

Animal Life

Student: Kryspin Marcysiak

Supervisor: Faisal Saeed

BIRMINGHAM CITY UNIVERSITY

BSc (Hons) Computer Science

Submission – 05/20

Kryspin Marcysiak

i. Abstract

Animal Welfare is a topic researched by many but not many solutions are making a noticeable positive impact. This report details the research, design and development of a solution called Animal Life. By the use of a mobile phone, a user can create sighting reports of endangered animals which will be attended to by appropriate teams. Adoption advertisements can be viewed which can reduce the fill within animal shelters creating more possible homes for endangered animals. Providing users with a pet need tracker so a pet never misses another appointment. All combining into one mobile application aiming to make a large impact.

ii. Acknowledgements

Throughout the journey of this project, I would like to express my deepest gratitude to my supervisor, Faisal Saeed who provided me with overall support.

Kryspin Marcysiak

iii. Contents

Contents

1Introduction	12
1.1Problem Definition	12
1.2Scope	13
1.3Rationale	13
1.4Project Aims & Objectives.....	14
2Literature Review	15
2.1Literature Review.....	15
2.2Review of Literature.....	16
2.2.1Stray Animals	16
2.2.2Education & Awareness	17
2.2.3Importance of UI & UX Design	19
2.2.4Importance of Pet Adoption	20
2.2.5Pets & Mental Health	21
2.3Existing Systems	22
2.3.1Helpimal.....	22
2.3.2Petsi.....	23
2.3.3Pet Health Tracker - Dog Cat	23
2.3.4Existing Systems Conclusion	25
2.4Summary.....	25
3Project Design & Methods.....	26
3.1Introduction.....	26
3.2Methodology	26
3.3Specification	27
3.3.1Limitation & Options	27
3.3.2User Requirements	28
3.4Data Persistence (Database).....	29
3.4.1Amazon Web Services	29
3.4.2Microsoft Azure	30

Kryspin Marcysiak

3.4.3Google Firebase	30
3.5Concept Solution	31
3.6Diagrams	32
3.6.1Database Diagram	32
3.6.2Use Case Diagram	33
3.7Wireframing	34
3.7.1Mobile Medium Fidelity	34
3.7.2Web Medium Fidelity	36
4Implementation	38
4.1Introduction	38
4.2Selected Data Persistence	38
4.3Firebase Configuration	38
4.3.1Authentication	39
4.3.2Firestore	40
4.3.3Storage	40
4.4Mobile Application	41
4.4.1Firebase Setup	42
4.4.2Main Activity	43
4.4.3Sign In & Sign Up	45
4.4.4Report	48
4.4.5Adopt	51
4.5Web Application	55
4.5.1Setting Up Connection	55
4.5.2Base Head Content	56
4.5.3Logon Page	56
4.5.4Side Navigation Menu	59
4.5.5Dashboard Page	60
4.5.6Report Page	62
4.5.7Adoption Page	64
4.5.8Educational Page	72
5Evaluation	73
5.1Functionality Testing	73

Kryspin Marcysiak

5.1.1Report.....	73
5.1.2Adoption Advert.....	75
5.2Application Feedback	77
5.2.1Feedback Criteria.....	77
5.2.2Feedback Results	77
6Conclusion	78
7Recommendations For Future Work.....	79
8Appendix	80
8.1Appendix 1	80
8.2Appendix 2	80
9References.....	82

iv. Glossary

Strays – Animals which are strayed/ homeless

AWS – Amazon Web Services

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

v. List of Figures

Figure 1: Process of educating pet owners	18
Figure 2: Helpimal Report Page Figure 3: Helpimal Report Options	22
Figure 4: Petsi Feed Figure 5: Petsi Happy Trail	23
Figure 6: Dog Cat Profile Figure 7: Dog Cat Premium	24
Figure 8: RAD Model Visualisation	26
Figure 9: Block Diagram Describing Stages of Project	31
Figure 10: Database Diagram	32
Figure 11: Use Case Diagram	33
Figure 12: Landing Page Wireframe	34
Figure 13: Main Wireframe	34
Figure 14: Login Wireframe	36
Figure 15: Dashboard Wireframe	36
Figure 16: Report Wireframe	37
Figure 17: Adopt Wireframe	37
Figure 18: Educational Wireframe	37
Figure 19: Firebase Authentication	39
Figure 20: Authentication Confirmation	39
Figure 21: Firestore Rules	40
Figure 22: Storage Rules	40
Figure 23: Authentication SDK	42
Figure 24: Firestore SDK	42
Figure 25: Storage SDK	42
Figure 26: Current User Authentication	42
Figure 27: Shared Preferences	43
Figure 28: First Run	43
Figure 29: Current User	43
Figure 30: Check Login State	43
Figure 31: Landing 1	44
Figure 32: Landing 2	44
Figure 33: Landing 3	44
Figure 34: Landing 4	44
Figure 35: Landing 5	44

Kryspin Marcysiak

Figure 36: Homepage	44
Figure 37: Navigation Menu	44
Figure 38: Sign Up Class	Figure 39: Sign Up Page 45
Figure 40: Sign In Code	Figure 41: Sign In Page 46
Figure 42: Sign Up Test	Figure 43: Account Confirmation 46
Figure 44: Password Reset Code	Figure 45: Password Reset Alert 47
Figure 46: Email Template	47
Figure 47: Report Spinners	48
Figure 48: Gallery Permission	48
Figure 49: Report Data Preparation	49
Figure 50: Report Data Insertion	49
Figure 51: Report Page Test	Figure 52: Report Confirmation 50
Figure 53: Report Firestore Values	50
Figure 54: Fetching Adoption Data	Figure 55: Adopt Page 51
Figure 56: Timestamp Calculation	51
Figure 57: Individual Advert Intent	52
Figure 58: Email Content	52
Figure 59: Adoption Request Firestore	53
Figure 60: Send Adoption Request	53
Figure 61: Adoption Pet Page	Figure 62: Email Option Figure 63: Preset Email Content 54
Figure 64: Configuration Values	55
Figure 65: HTML Connection	55
Figure 66: HTML Head Content	56
Figure 67: HTML Logon Body	56
Figure 68: HTML Logon Output	57
Figure 69: JavaScript Firebase Initialization	57
Figure 70: JavaScript Login Event	57
Figure 71: JavaScript Remain Logged In	58
Figure 72: JavaScript Invalid Input	58
Figure 73 : JavaScript No Access	58
Figure 74: JavaScript Login Successful	58
Figure 75: HTML Navigation Menu	59
Figure 76: Menu CSS 1	Figure 77: Menu CSS 2 Figure 78: Menu Output 59
Figure 79: Google Analytics Setup	60
Figure 80: Google Analytics Tag	60

Krystin Marcysiak

Figure 81: Google Analytics Report	60
Figure 82: Google Analytics iframe	60
Figure 83: Dashboard Analytic Issue	61
Figure 84: HTML Report Table	62
Figure 85: JavaScript Report Import	62
Figure 86: JavaScript Report Query	62
Figure 87: JavaScript Report Query	63
Figure 88: JavaScript Report Content	Figure 89: JavaScript Report Rows
Figure 90: JavaScript Report Output	63
Figure 91: HTML Adoption Table	64
Figure 92: HTML Adoption Popup 1	Figure 93: HTML Adoption Popup 2
Figure 94: HTML Adoption Button	64
Figure 95: JavaScript Adoption Import	65
Figure 96: JavaScript Adoption Initialization	65
Figure 97: JavaScript Adoption Document Fetch	65
Figure 98: JavaScript Adoption Referencing 1	65
Figure 99: JavaScript Adoption Referencing 2	65
Figure 100: JavaScript Adoption Row Fill 1	66
Figure 101: JavaScript Adoption Row Fill 2	66
Figure 102: JavaScript Adoption Button	66
Figure 103: JavaScript Adoption Output	67
Figure 104: JavaScript Adoption Button	67
Figure 105: JavaScript Adoption Conversion	67
Figure 106: JavaScript edit ad updateDoc	68
Figure 107: JavaScript Edit Ad	68
Figure 108: JavaScript Edit Ad Error	68
Figure 109: JavaScript Adoption Overlay	69
Figure 110: JavaScript Popup Form	69
Figure 111: JavaScript Popup Code	70
Figure 112: JavaScript Popup Test	70
Figure 113: JavaScript Popup Confirmation	71
Figure 114: JavaScript Added Successfully	71
Figure 115: Educational Row	72
Figure 116: Educational Popup	72
Figure 117: Educational Error	72

Kryspin Marcysiak

<u>Figure 118: Educational Added Successfully</u>	72
<u>Figure 119: Report Test Image</u>	73
<u>Figure 120: Report Database Update</u>	74
<u>Figure 121: Record added into table</u>	74
<u>Figure 122: Adoption Test Image</u>	75
<u>Figure 123: Adoption Added Into Firestore</u>	75
<u>Figure 124: Adoption Page</u>	76
<u>Figure 125: Unique Adoption Page</u>	76
<u>Figure 126: Email Test</u>	76
<u>Figure 127: Email Arrival</u>	76

<u>Table 1: Table stating each theme accompanied by the keywords used to find relevant literature</u>	15
<u>Table 2: HTML Login Results</u>	58
<u>Table 3: Test Report Data</u>	73
<u>Table 4: Adoption Test Data</u>	75
<u>Table 5 User Testing</u>	77

1 Introduction

This project focuses on developing a mobile application within Android Studio which can provide a solution towards the rising population of stray animals. Focusing on a sighting reporting system which enables users to create reports on an animal which may be in danger will enable administrators to notify a suitable team, this increases efficiency of rescue teams. Project aims, themes, resources, and methodologies will outline key aspects of the report. The report will outline literature surrounding relevant topics for a detailed literature review. The design and implementation of the solution will be included in this report. Finally, concluding with a testing stage and a discussion of the results, followed by a conclusion of the project.

1.1 Problem Definition

Animal welfare but in particular stay animals, has been a rising issue for decades and does not seem to face a decline anytime soon. The European Union has estimated that the population of stray companion animals such as cats and dogs is around 100,000,000, this is solely within the European Union's member states (ESDAW). Stray animals are endangered, they have limited access to veterinary services to receive vaccines, increasing health and safety risks for animals and humans (Lappin, 2014). In addition to this, animal rescue shelters are overpopulated, this results in other strays not being able to be taken off the streets due to sheltered animals not being taken into new homes. There are plenty of factors contributing to the over population of strays worldwide, a major factor is that owners are not taking the necessary precautions such as sterilization which prevents an animal from continuing the reproductive cycle, are not being taken. Therefore, when owner's neglect their animals, those animals become strays and are open to reproduce, in-turn increasing the population.

1.2 Scope

The project scope is to create a system that will make an effect on the stray animal problem. A mobile application allowing users to make reports, view adoption adverts and receive educational material to raise awareness will be beneficial. In addition, a management website allowing administrators to handle the information coming from and to the mobile application should be developed. On the other hand, there are limitation such as the data security regarding the general data protection act. Application will only be suited for the United Kingdom due to time limitations.

1.3 Rationale

This project will benefit animals and a wide range of people that can benefit from this application. Users being able to create sighting reports of stray animals will enable rescue teams to have an easier experience which gathering stray geo-graphical data to rescue stray animals off the street. The adoption feature will enable users to adopt a stray animal from an animal shelter, in-turn creating availability for more strays to be taken into care. Educating users with animal welfare material will improve the quality of life of animals and make an impact on other issues such as overpopulation, health & safety risks and more.

1.4 Project Aims & Objectives

The aim of this project is to develop a mobile application allowing users to make a positive impact on animal welfare. The mobile application will aim to address in particular problems such as stray animals, over-population within animal shelters, poor quality of life for animals. The content of the mobile application should be handled using a web portal only accessible by authorised users like admins.

Project Objectives:

- The mobile application should be fully functional, following design laws to increase user retention and overall satisfactory experience. To be successful, the mobile application will not crash, will function according to the design as well as be linked to a database.
- The mobile application should have a reporting feature allowing users to make sighting reports of animals that may be strayed, distressed, injured etc. To successfully function, the application should allow a user to report key information such as type of animal, the state the animal is currently in, a location and an image.
- The mobile application should have an adoption feature allowing users to view adoption adverts. If successful, once the user is satisfied to adopt an animal, then the user should be directed to contact the relevant organisation.
- The mobile application should spread awareness to users with the purpose of improving an animals life. A blog would be an adequate feature. For this feature to be successful, the user will be able to view recent blog posts which are made via admins using a web page.
- The mobile application's data such as adverts, reports, user information should all be controlled via a dashboard on a web page accessible to authorised users such as admins. If successfully functional, the web portal will enable admins to log-in, view animal reports, post adoption advertisements and more.

Kryspin Marcysiak

2 Literature Review

2.1 Literature Review

This section of the report will cover multiple literature which has been researched for this project. The key themes of the literature were discussed prior to the review and summary of the literature.

Theme	Keywords
Stray Animals	Animal welfare, Pollution Control, Stray Animal Importance, Diseases in stray animals,
Education & Awareness	Animal Welfare Awareness, Animal Welfare Education, Importance of Awareness,
Importance of UI & UX Design	Importance of UI Design, UI Design Methods, Colour Theory, Fitt's law, Hick's Law, Interface Design Principles,
Importance of Pet Adoption	Importance of Pet Adoption, Shelter Statistics,
Pets & Mental Health	Pets & Mental Health, Metal Health, Domestic Pets, Human's Best Friend

Table 1: Table stating each theme accompanied by the keywords used to find relevant literature

2.2 Review of Literature

Relevant literature will be reviewed in this section, providing further knowledge, and understanding of the rising stray animal issue.

2.2.1 Stray Animals

Stray animals are a term used for free-roaming animals without a home making them vulnerable to threats and diseases (Abubakar, 2021). These animals become homeless mainly from being stranded by their owners, which may begin a cycle of an animal becoming pregnant and increasing the stray animal population in the world. This issue has been addressed by international animal welfare organisations by providing homes in forms such as shelters and adoption methods. Although this has provided benefits, the WHO stated that the stray animal population is rising again and as of 2023 the population of stray dogs is approximately 200 million and even more stray cats (Four-Paws, 2023). The longevity and life span of stray animal is dependent on the environment and several key factors such as access to water, access to food, access to shelter.

Stray animals are unfortunately a possible threat to humans and other animals as they could attack if distressed. Stray dogs can be easily distressed due to trauma lived by the kennel and any form of contact can trigger a sense of danger, therefore resulting in an attack. These dogs may hold zoonotic diseases that can transmit onto other people and other animals through bites and scratches (Abubakar, 2021). Dogs have always provided support for a human such as gate protection, drug sniffing, a human's non-human best friend and more (Abubakar, 2021). So why are people in the modern age disregarding their importance and leaving them stranded in the wild? Therefore, animal control is a very vital method to reduce the risks from bacterial infections and diseases. Animal control techniques vary depending on the country in which the stray is in but the primary method of animal control within Europe is "stray dog collection" and which are then kept in a shelter (Louisa, 2023). Another common animal control method is the conduction of neutering of animals resulting in them being unable to reproduce, this may prevent one to ten additional animals being born into the world of homelessness (Louisa, 2023).

There are two methods of animal control which are not humane and should not be conducted and that is poisoning and bulleting (Svanhild, 2022). These will cause a stray to suffer from hours to months slowly dying away similar to how pest control deal with rats or

Kryspin Marcysiak

via a bullet shot by the local organisation. These methods are mainly conducted in south American countries (Svanhild, 2022).

2.2.2 Education & Awareness

Owner education is a vital requirement when owning an animal as animals are not “simple” entities to maintain. An owner is required to understand the needs of an animal, in example; the type of injection required, how often vaccinations are necessary, types of food required and more. Whilst irresponsible cold-hearted owners are a factor to poor animal quality of life, under-educated owners are also a cause of animal under-care, abandonment, and a bad quality of life (Louisa, 2023). School teaching on animal welfare has been arising in schools across Europe (Svanhild, 2022). Although school teaching and pet ownership support available, the seems not to be decreasing and instead increasing as 2022 had a 23% increase from 2021 and 2023 had an increase of 2022 by 8% of reported cases in stray animals, animal abuse psychological and physical abuse (Kenny, 2023). This once again is due to numerous factors, but more people are unable to afford animals due to the current cost of living (Kenny, 2023).

Animal owners that own domestic animals are required to have knowledge of animal upkeeping and there are organisations such as WHO and RSPCA which have materials available online or in-person workshops, but this is not “convenient” to the average person as mobile applications would be more applicable for on-demand access. For example, RSPCA offers online activities that can guide animal owners with short-handed guides on “making a rabbits life better” or “baking treats for your companion to improve their life quality”. Life of quality for all animals is important as the media portrays them to be an integral part of a family (Izzie, 2019). This is a very important feature available to the public allowing them to gain further understanding of animal welfare. Mobile applications are vital in general education due to the ease of access, and it results in users having a 20% higher probability of staying on the application rather than a web-based application (Athanasios, 2017).

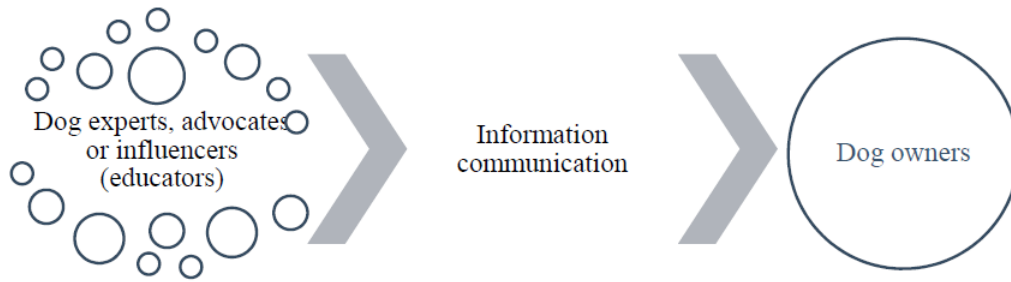


Figure 1: Process of educating pet owners

Whilst there are multiple laws, rules and regulations worldwide which cover animal care, welfare. A vital regulation is the microchip regulation allowing an animal to be found if lost via a unique associate number similar to a VIN identifier on vehicles. This results in a lost dog being identified and rescues back to their associated owner (Izzie, 2019). Education on laws and regulations therefore is a key in animal up-keeping, in 2022 30% of lost animals made their way back to their owner due to an active microchip (Svanhild, 2022).

2.2.3 Importance of UI & UX Design

Mobile application design methods are very important in mobile application development as they can increase use retention and improve experience (Mariam, unknown). This can improve the success rates for the applications purpose, for example, if the application is designed using various methods the user will be more likely to stay and learn about animal welfare which will improve an animals quality of life.

A user interface is the front end of an application otherwise referred to as UI. Therefore, the visualisation of this vital for user experience and is supported by several methods. Jakob's law is crucial in interface design, this requires mobile/web pages to be consistent and having similar visual design, workflow, and information (Jakob, 2020). This will facilitate a better user experience improving their probability of staying on the application for longer. Fitts's Law is another vital principle in interface design, this involves having a clear interface minimising user mistake (Xiangqian, unknown). Having well-spaced-out elements avoiding interactable buttons being too close together will reduce the probability of a user making a mistake that may lead to them exiting the application and ending their session (Xiaojun, 2013). Hick's Law is another principle used when addressing user experience and it requires choices and data to be easily understood allowing users to have faster reaction time and less brain strain. This addresses user accessibility creating a safer and easier application for types of users (Fakhri, 2022). Colour theory is the term used when deciding the colour pallet for an application. This is important as colours are related to emotions, reactions, feelings, thoughts and more. The colour red will likely not correspond to the theme of positivity on the other hand, green, blue might (Nasr, 2014). The colours orange, green, blue, and pink are useful when creating applications as they symbolise positivity and help absorb new ideas (Nasr, 2014). On the other hand, colours should not be clashing and should show smooth contrast to remove eye strain and address accessibility requirements such as epilepsy. All interface design principles and methods are key when designing an application to ease a user's experience which will increase the probability of application purpose being executed (Jakob, 2020).

2.2.4 Importance of Pet Adoption

Pet adoption rates decreased from 2012 to 2020 every year by 5% for unknown reasons (Jeffery, 2021). On the other hand, following the Covid-19 pandemic the pet adoption rates increased by 250% due to mental health support (Jeffery, 2021).

Upon conducting a literature review a very powerful journal by a group of researchers has provided vast evidence for this theme. Within the United States of America 10 dogs per 1000 residents are admitted to dog shelters annually (Lauren, 2018). This is a rising issue worldwide because shelters are running out of space which leads to shelters no longer taking in animals, reducing the probability of a stray animal being taken into a safe shelter. Therefore, pet adoption is a vital factor to shelters being freed up from space allowing other animals to be taken in (Lauren, 2018).

Pet owners typically address matters when considering pet adoption such as, was the pet homeless? Was the pet hurt? Does the pet have a shorter lifespan? (Goitom, 2013). This is something that is typically addressed in dog shelters and owners pick out characteristics mentioned previously. This is particularly unfair as a hurt animal does not decrease the love value of an animal (Lauren, 2018). Descriptions which are used when describing a pet that is up for adoption is very important as it is the information provided to the user and can influence decisions (Amir, 2022). A choice of image is also very vital to influence the user's choice as it can provide more visualised context to their interest (Amir, 2022).

2.2.5 Pets & Mental Health

Domestic pets can have a large influence on a person's mental health, this is due to pet's providing companion support. Pet owners gain higher levels of mental health wellbeing allowing them to cope better with adverse situations, an example of this is during Covid-19's Lockdown (Dasha, 2021). This is because humans are capable of developing a deep bond with animals as they help a human develop characteristics such as responsibility, nurturance, and an adherence to the daily schedule (Odean, 2013).

A domestic pet increasing a pet owners mental wellbeing is maximised once the pet is in good health. This means that the animal is loving of the owner due to the owner treating the animal with fairness and a loving heart (Dasha, 2021). Animals are also good companions for children as they can provide support with activities and even learning (Dasha 2021).

A study conducted of 309 students of whom 140 owned pets, 102 has owned pets once in their life and 67 have never owned pets was tested on anxiety, depression, behaviour, and other mental health processes (Odean, 2013). The study shown lower stress levels and calm levels for the 102 students which owned pets which implies that pets can stabilise mental health processes. Veterans are typically accompanied by fellow ferried friends as they can provide support towards PTSD that they may be suffering from (Steohen,2013). This ranges from outside support to home support where animals can help a veteran hug, put a blanket over them and show them love and attention which may be required (Terry, 2018). Domestic pets such as dogs are particularly great for mental health reason and have been for decades and are companions for humans from vast backgrounds, vast situations, and housing types (Odean, 2013).

2.3 Existing Systems

Throughout time, a wide range of systems have been developed to make a positive impact on animal welfare. There is a mobile application representing each objective of this project, this results in a user having to change applications for each intent. Below I will discuss existing systems found in the app store (IOS) in relation to the project objectives:

2.3.1 Helpimal

Helpimal is an animal reporting application in mobile form, allowing a user to report an animal in need of help. Using a user's current geo-location, the user can make a report by providing an image, animal type, category, additional information, and uses current location. Whilst a current location is helpful, sometimes a user might have to manually enter an address for a more accurate report which this application cannot do. Helpimal uses a modern design which contrasting colours, this creates a smoother and a lighter experience for the user which can lead to the increase of a user's retention.

Once a user makes a report, the report is public to other users on a map. This could imply that Helpimal wants users to work as a community to make a difference on the world. It is unclear if administrators forward reports to relevant rescue bodies as there is not a wide range of information available. In addition, the application is based in Turkey therefore the translation of the application description is not accurate.

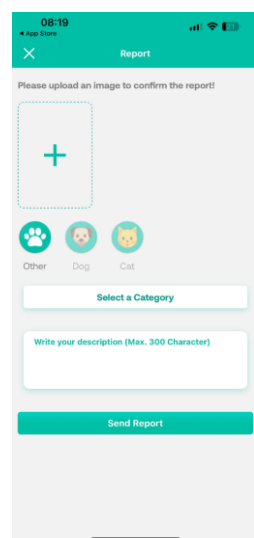


Figure 2: Helpimal Report Page

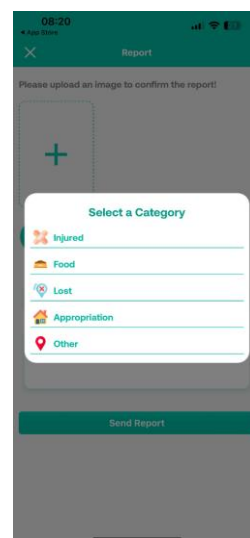


Figure 3: Helpimal Report Options

Kryspin Marcysiak

2.3.2 Petsi

Petsi is a mobile application only available in Europe which allows users report animals as missing, found or up for adoption. The mobile application is based around a simple feed like design, this is a very common layout for advert-based applications such as Pets4Home. A user is able to expand more information about the advert and will be presented with the owner's contact information to provide a location or last known location of their pet. This application serves a purpose but is limited with features. When a user wants to report a pet as missing, the pet is only shown on the feed and is not passed onto appropriate authorities. It would be very beneficial to forward the missing animals information to the local rescue teams and adoption shelters which will increase the probability of their pet being found.

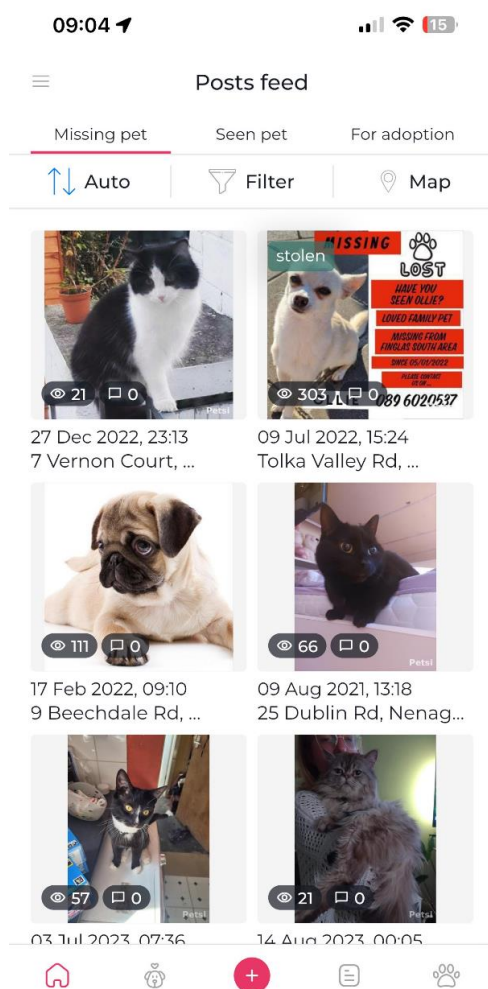


Figure 4: Petsi Feed

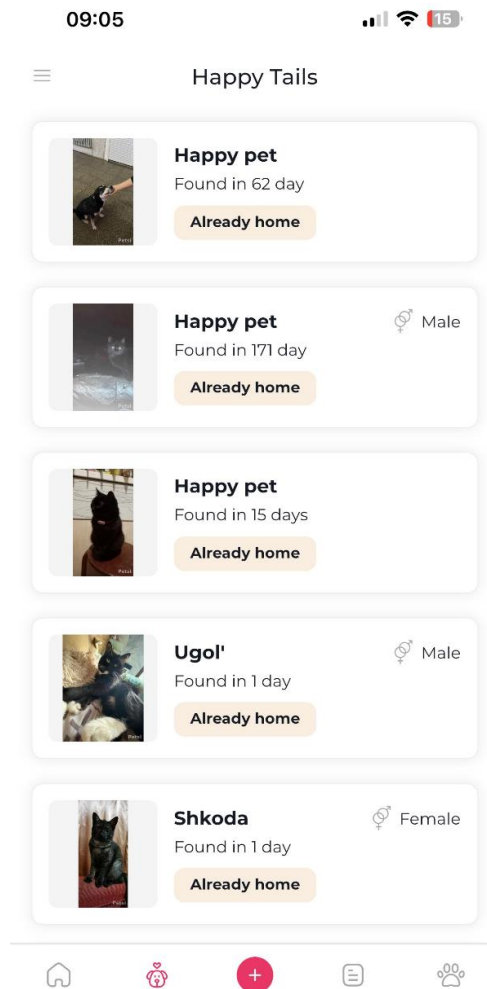


Figure 5: Petsi Happy Trail

2.3.3 Pet Health Tracker - Dog Cat

Keeping track of a pet's health is very important and the Dog Cat pet health tracking application does just that. The application prompts the user to create a profile for their pet and input base information such as name, age, photo. Dog Cat offers a calendar system which is used to keep track of the pets vaccinations, appointments and more. If the user has multiple pets with multiple calendars, then the main calendar will present the user with all events. In addition, the mobile application presents weekly animal facts to the user which can raise awareness, providing guidance to pet ownership.

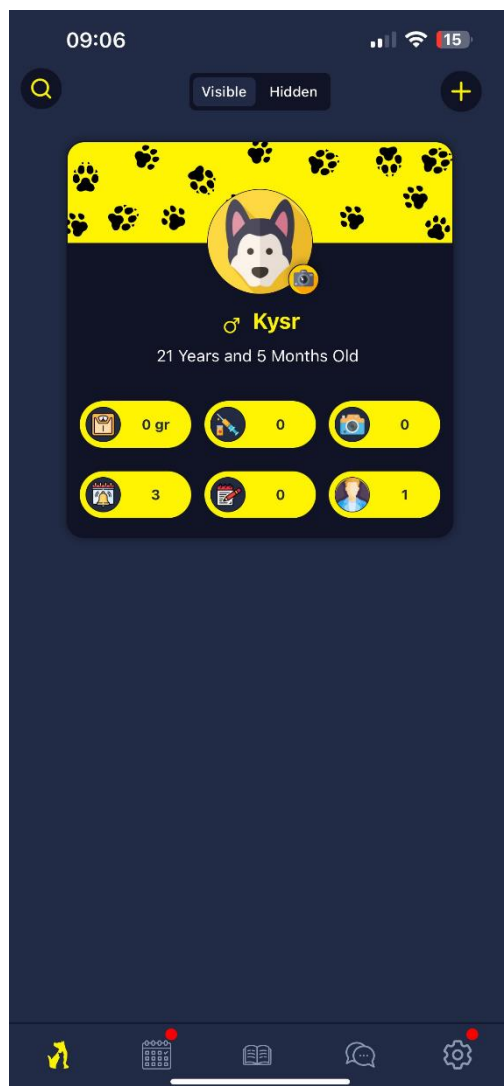


Figure 6: Dog Cat Profile

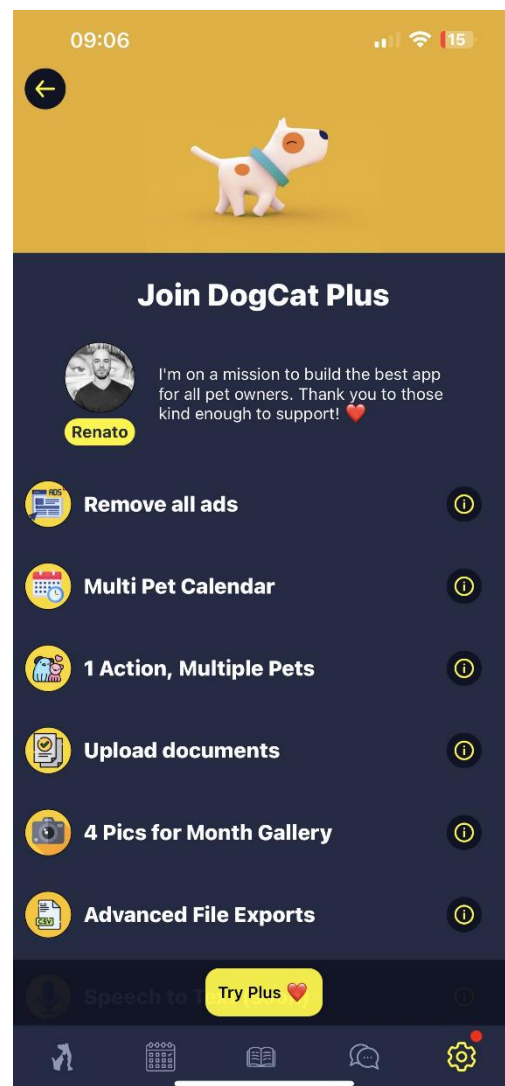


Figure 7: Dog Cat Premium

2.3.4 Existing Systems Conclusion

Upon researching existing mobile applications, it is clear that there are applications which address animal welfare, in particular my system objectives. On the other hand, these applications are singular and come in individual applications. This will result in the user having to switch applications to access each feature, more storage being used as there are three or more applications. It is important to maintain an application's ease of use which can improve a user's experience and increase their user retention. This provided key information which can improve the application design and development process.

2.4 Summary

To conclude this literature review, animals have a noticeable impact on humanity in ways of providing companionship, mental health support and more. Therefore, it is important that animals are provided support back by providing them with a quality life filled with love, care, and support. Stray animals are an issue rising worldwide that create an unsafe environment for other animals and humans due to diseases and distress that may result to an attack. This is caused by animals being stranded by careless owners and are homeless due to it creating an un-fair life for them. Pet adoption can reduce the shelter capacities allowing more stray animals to be taken into shelters and provided with basic necessities such as food, water, and shelter. Creating a cycle of positive affection.

With taking these matters into account it is important a solution is created in the form of a mobile application which will be discussed further later. When designing and developing this solution it is important that interface design principles are considered to create a good experience for a user which will maximise the probability of an effect on the problem of stray animals. Whilst also educating current and future pet owners on creating a better life for domestic animals.

Kryspin Marcysiak

3 Project Design & Methods

3.1 Introduction

This section of the report will explore methods, design principles and design methods which will be used for the final artefact. The literature review principles discussed previously, and further research will be used to complete this section and guide the artefact product. A methodology will be identified, following project limitations, user requirements, a concept solution, testing strategies, flow diagrams and a summary.

3.2 Methodology

The methodology which will be used to guide this project is the rapid application development (RAD) methodology. This is an appropriate methodology for this solution as it allows trial and error management. The design is constantly refined and tested by using interface design principles until satisfaction is met. Later leading to the construction of the solution which consists of implementation of the solution and the testing, it is only then finished once the product meets all the requirements. Therefore, this methodology is great for this solution allowing for trial and error, more efficient change cycles and a broader cycle.

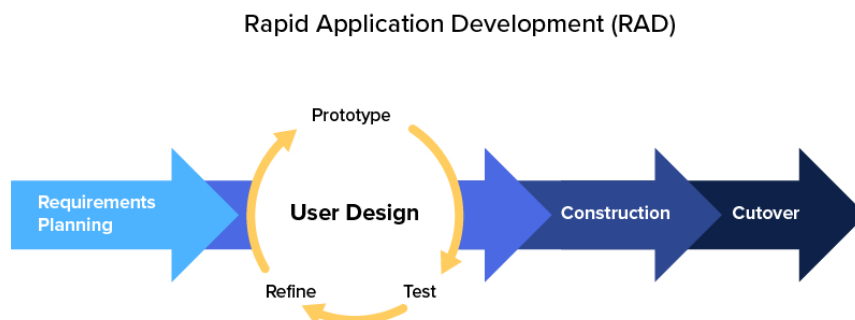


Figure 8: RAD Model Visualisation

3.3 Specification

3.3.1 Limitation & Options

There are numerous limitations for this solution and for the themes discussed previously. But all of these themes will have options and possible solutions that address the issues.

Stray Animals: Stray animals are the main focus of this report and the artefact therefore a solution for this will need to be created. There are a number of ways of maintaining the population of stray animals. The solution for this theme that will be used is the implementation of a mobile application using a sighting reporting service and an emergence report service that will be accessible to user's with admin access and organisations to monitor reports. This will include visualisations such as area report hotspots showing the area where interest and action needs to take place. The limitations to this are the scale in which I will be conducting this in, focusing on a smaller scale will be more adequate and an example of this can be just focusing on the United Kingdom or Europe. In addition to this I will require further understanding and knowledge of mobile application development to accomplish this solution, as well as knowledge of Artificial Intelligence. The admins and organisations will likely be stationary when conducting reports and examining them therefore a web-based portal will be applicable for this. This will require website hosting and linking a mobile application, website to a database. This is usually done by the use of MS SQL Servers which is entry level and makes this the perfect option. Android Studio, HTML, MS SQL, PHP, Python and JavaScript will be used to develop this.

Education & Awareness: It is important that education and awareness is spread across the application to improve a pet owners knowledge which in turn improves a pets quality of life by having a more understanding owner. This can be implemented by including guides, tips & tricks, articles on pet ownership. This can be accomplished by the knowledge of mobile application development and knowledge of the matter. Admins will require access to do this.

Kryspin Marcysiak

Importance of UI & UX Design: This is a very important theme which has to be addressed when designing a mobile application and a web application. This is so that the user has a satisfactory experience so that their purpose can be fulfilled by the use of the mobile application. In addition to this, the web-based application will have to follow the same principles so the session can conclude in a lower exit rate and an easier experience. This can be done by using numerous principles mentioned in the literature review such as Fitts's Law, Hick's Law and more that I have yet to research. Furthermore, accessibility features need to be implemented to create a fair accessible experience for all users.

Importance of Pet Adoption: Pet adoption is an important step that needs to be implemented so existing shelters can reduce capacity allowing them to take more stray animals off the dangerous streets. This can be done by implementing a pet adoption feature that will show good quality information, pictures following the interface design principles. This will require knowledge of mobile application development and allow admins to view and track traffic by showing analytics and statistical reports. All requirements for this solution are free-of-charge allowing for lower costs.

3.3.2 User Requirements

The following user requirements for the solution are in a bullet pointed list below based off the literature review, in addition to identifying the limitations and options and themes for the project:

- An accessible friendly mobile application with quality information about the issue.
- A stray animal sighting reporting feature which be stored and tracked.
- An account system allowing a user to manage account and store current pets resulting in better health tracking
- Education & awareness spreading feature allowing admins to spread useful information.
- An adoption service allowing users to view pictures and key information about dogs looking for a home.
- A webpage for admin and organisations that will allow them to monitor statistics on stray animals and other key features.
- The mobile application and web-application will both require to be linked to a database.

Kryspin Marcysiak

3.4 Data Persistence (Database)

The choice of data persistence, also known as a database, is a very crucial choice. This is due to each data persistence holding unique benefits, each data persistence also will provide limitations to the application. It is important to discuss the main choices of data persistence and select the most appropriate one.

3.4.1 Amazon Web Services

Amazon Web Services short for AWS provides tools and services to create a secure cloud environment. To tailor towards this Animal Life project, a wide range of features would be required. Most features are paid for.

- Amazon Cognito – This feature would ensure that the mobile application enables users to sign in, sign out and access mobile applications using secure user authentication.
- Amazon API Gateway – Using gateways, this tool ensures a connection between the mobile application and the web application. In addition, this feature provides a role function which will require a user to be an ‘admin’ to access the web application.
- Amazon DynamoDB or RDS – Both tools provide a database service. Whilst DynamoDB is not controlled via SQL, provides light and high scalability, RDS provides a database controlled via SQL which most programmers are familiar with.
- Amazon VPC – Using private network control which can restrict inbound and outbound traffic. This is useful as it offers the ability to limit regional traffic to and from the systems.

3.4.2 Microsoft Azure

Microsoft Azure like AWS provides tools and services to make systems come to life. Alike Amazon Web Services, most features require payment.

- Azure Active Directory – This is the user authentication tool which provides application access to users with valid logon credentials.
- Azure App Services – This feature provides highly scalable web application hosting.
- Azure Functions – Providing business logic between a mobile application, a database and a web application, Azure Functions establishes a connection between all deployed applications.
- Azure SQL Database - This is a database in an SQL language

3.4.3 Google Firebase

Google Firebase is a widely accessible project management system. Providing a wide range of tools to make a system come to life at a low cost or no cost, this is a programmer's main choice for project development.

- Firebase Authentication – There is a wide range of user authentication features, allowing users to sign up using emails, phone numbers and third-party companies such as Apple, Facebook and more.
- Firebase Firestore – Using collections which hold documents with unique IDs, Firestore enables for a smooth database experience.
- Firebase Storage – Using folders which hold images uploaded by admins or users can have links assigned. If a link is opened within an application, the image will display to the user.
- Firebase Analytics – This tool tracks the access of applications, user interactions, read and writes. Using Google Analytics, data can be visualised on the web application.

3.5 Concept Solution

The concept solution is a mobile application titled “Animal Life” which will allow a user to make stray animal sighting reports, emergency support for distressed animals that may be a threat to the public or itself. In addition to this, the user will be able to create an account and manage owned pets including health information as well as report their animal as lost or other animals as found. A pet adoption features will be included to provide shelter population control by showing available pets for adoption. A web portal will also be implemented for admins and organisations allowing them to view statistics and stray reports. In addition to this, admins will post articles and educational reports which will raise awareness to pet owners on animal wellbeing. All will be designed by the use of principles to create a satisfactory and an accessible experience for the user. Design mock-ups will be created using Figma. Mobile application will be developed in android studio (Java) and the web portal will be developed using HTML, CSS, JavaScript in a compiler. The database will be handled using Firebase due to its versatility.

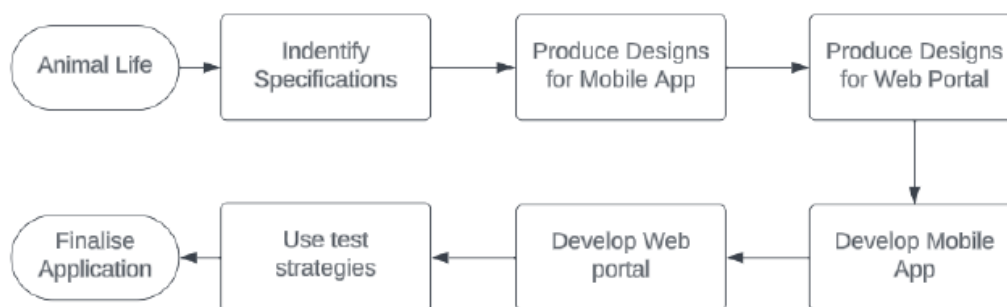


Figure 9: Block Diagram Describing Stages of Project

3.6 Diagrams

3.6.1 Database Diagram

Every time an authenticated user (admin and user) creates a report, posts an adoption advertisement, educational material then their user ID will be included in the firebase document. When a user adds a pet to their account, it will be associated with their Firestore user ID which is a foreign key.

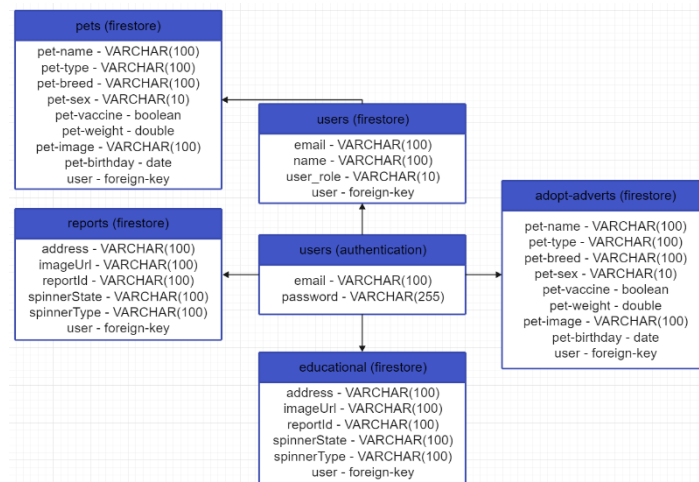


Figure 10: Database Diagram

3.6.2 Use Case Diagram

A use case diagram presents the relationship between main features. A user can make a report which will enable an administrator to view the report. In addition, a user can add a pet which will be associated with their account. An administrator is able to post adoption advertisements and educational material for the main user to view.

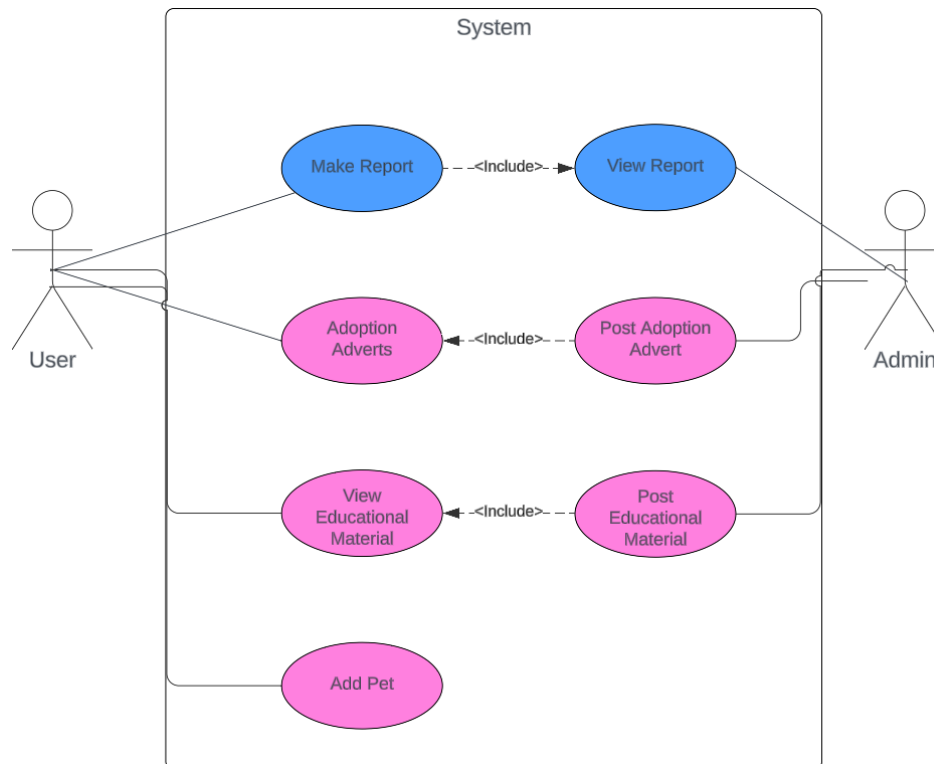


Figure 11: Use Case Diagram

3.7 Wireframing

Upon taking all previous content but prior to developing the mobile application and the web application, it is important to construct a design. There are a lot of factors to consider such as Jakob's law, Fittz's law and more, these can be found in [2.2.3](#).

3.7.1 Mobile Medium Fidelity

There are multiple factors to consider when developing a design for a mobile application. After considering the project requirements and previous diagrams, a design can be created. A landing page is a very common feature within mobile applications, a study highlighted that applications with multiple landing pages resulted in higher conversions (Helen, 2018).

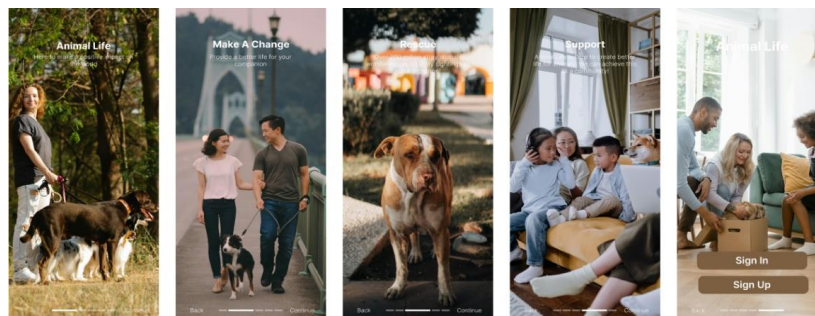


Figure 12: Landing Page Wireframe

A set of four introductory landing pages are key as they introduce the user to the application and its key features. Using images to enhance the positivity of the application, a progression indicator to clearly tell user what page they are on can increase the sense of curiosity about what could be next. Finally, a logon page with two buttons in the applications theme.

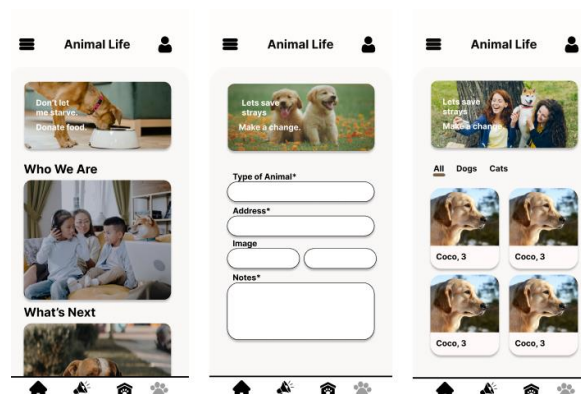


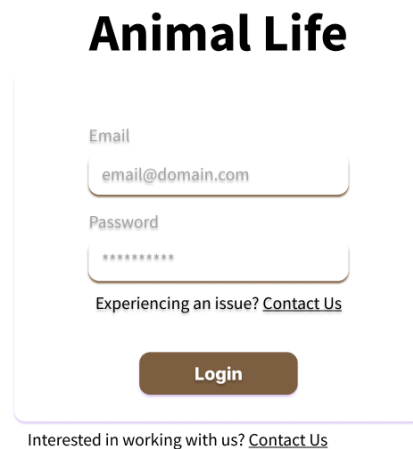
Figure 13: Main Wireframe

Using contrasting colours and a modernised theme, interactable materials within the mobile application are clear to the user. As this is just a design, it is clear that changes will be made due to limitations presented by Android Studio.

3.7.2 Web Medium Fidelity

After taking project requirements into consideration, it is clear what features are required. A dashboard themes web application is most appropriate for this project. There will not be a low-fidelity design for the web application as there is a solid idea of the layout.

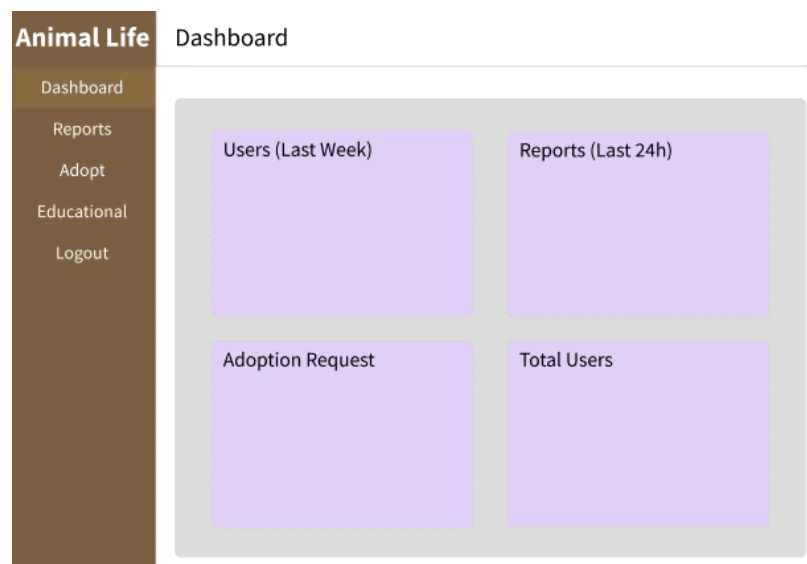
A simple logon page allowing a user to login. No register page is needed as administrators would manually authenticate others. The brown colour theme is followed for consistency.



The login wireframe for 'Animal Life' features a central white box with a brown border. At the top, the title 'Animal Life' is displayed in a large, bold, black font. Below the title, there are two input fields: 'Email' with the placeholder text 'email@domain.com' and 'Password' with a masked password '*****'. A link 'Experiencing an issue? [Contact Us](#)' is positioned below the password field. A brown 'Login' button is centered at the bottom of the box. Below the box, a footer line reads 'Interested in working with us? [Contact Us](#)'.

Figure 14: Login Wireframe

The main page of the web application will be the dashboard page, providing visualisations of total reports, adoption adverts and more. The side navigation menu follows the colour theme of the other pages and the mobile application.



The dashboard wireframe for 'Animal Life' consists of a brown sidebar on the left and a main content area on the right. The sidebar contains the 'Animal Life' logo at the top, followed by a 'Dashboard' header, and a list of navigation items: 'Dashboard', 'Reports', 'Adopt', 'Educational', and 'Logout'. The main content area has a 'Dashboard' header and a grid of four purple boxes. The top-left box is labeled 'Users (Last Week)', the top-right box is labeled 'Reports (Last 24h)', the bottom-left box is labeled 'Adoption Request', and the bottom-right box is labeled 'Total Users'.

Figure 15: Dashboard Wireframe

Kryspin Marcysiak

A page showing all animal reports and clearly stating the state of animal. A button allowing a user to view the image attached within the report, as well as a PDF button allowing an admin to forward the report to the appropriate body.

Animal Life		Reports			
Dashboard					
Reports					
Adopt					
Educational					
Logout					

ID	Pet Type	Pet State	Address		
1	Dog	Stray	10 Riverwood	Image	PDF

Figure 16: Report Wireframe

The adopt page displays all active adoption advertisements, as well as an edit button to alter the information of the advertisement. A button to add an adoption advert has also been implemented.

Animal Life		Adopt				
Dashboard						
Reports						
Adopt						
Educational						
Logout						

Add Animal for Adoption

ID	Name	Breed	Sex	Weight	Vaccine	
1	Chop	Dog	Male	42	True	Edit

Figure 17: Adopt Wireframe

Similar to the adoption page, the educational page will present active advertisements accompanied by an edit button. A button to post educational material will enable authorised users to make posts.

Animal Life		Educational		
Dashboard				
Reports				
Adopt				
Educational				
Logout				

Post Educational Material

ID	Title	Author	
1	Best Control...	Kryspin	Edit

Figure 18: Educational Wireframe

4 Implementation

4.1 Introduction

This section of the report will provide detail on the development of the mobile application and web application. Using Android Studio to create the mobile application and Visual Studio Compiler to develop the web application.

4.2 Selected Data Persistence

Throughout researching types of data persistence applicable to this project, it is clear that Google Firebase is the best choice. This is because the tools required by this project are cost-free. In addition, Firestore offers lighter data manipulation which will ensure that the development and syncing of the applications is not as complex. This will increase the time available during this implementation stage.

4.3 Firebase Configuration

Majority of Firebase configuration will occur when developing the applications. The base step to setting up firebase is to create a project with the name of "Animal Life", enable Google Analytics to enable future data analysis and keep track of interactions for security purposes.

4.3.1 Authentication

The authentication tool is used to integrate the login credentials of a user. This enables a user to create an account, sign into their account, reset their password, change email and more.

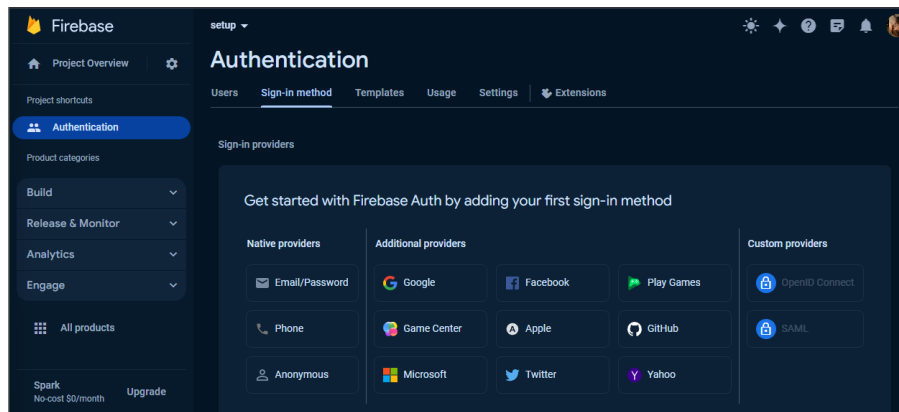


Figure 19: Firebase Authentication

This project will use the email and password sign-in method. The method should now be enabled as shown below.

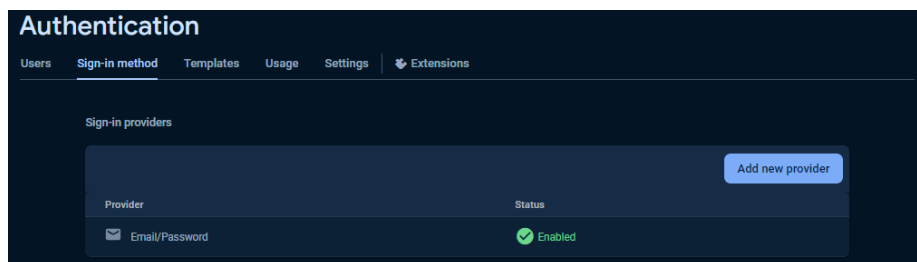


Figure 20: Authentication Confirmation

4.3.2 Firestore

The most appropriate database tool which will be used within this project is Firestore, it will use collections to store documents with unique IDs, this will ensure a lighter development process. Upon setting up a Firestore database, the most important step is updating the Rules.

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {

    // This rule allows anyone with your Firestore database reference to view, edit,
    // and delete all data in your Firestore database. It is useful for getting
    // started, but it is configured to expire after 30 days because it
    // leaves your app open to attackers. At that time, all client
    // requests to your Firestore database will be denied.
    //
    // Make sure to write security rules for your app before that time, or else
    // all client requests to your Firestore database will be denied until you Update
    // your rules
    match /{document=**} {
      allow read, write;
    }
  }
}
```

Figure 21: Firestore Rules

4.3.3 Storage

Images will be uploaded and accessed in numerous ways. Firebase Storage will allow this project to do just that. Once an image is uploaded, it will be assigned a hyperlink which can be accessed via the application. It is important to ensure adequate rules are entered

```
1 rules_version = '2';
2
3 // Craft rules based on data in your Firestore database
4 // allow write: if firestore.get(
5 //   /databases/{default}/documents/users/{request.auth.uid}).data.isAdmin;
6 service firebase.storage {
7   match /b/{bucket}/o {
8     match /{allPaths=**} {
9       allow read, write: if false;
10    }
11  }
12 }
```

Figure 22: Storage Rules

4.4 Mobile Application

This mobile application will be developed within Android Studio and the main programming language used within this are Java. Prior to creating the application, a setup of the development environment was necessary due to system limitations. I was required to deactivate a large number of plugins within the environment to increase performance, Android Studio is a very intense applications as it simulates a mobile phone locally.

All of the following plugins which will not interfere with the development of the application:

- YAML
- GitHub
- Mercurial
- Subversion
- Android APK Support
- Android NDK Support
- App Links Assistant
- ChangConfig
- ClandFormat
- Copyright
- Device Steaming
- Google Cloud Tools
- Task Management

The development process of the applications layout (visualisation) will not be included, only the logical process of the main features will be covered within this report.

Kryspin Marcysiak

4.4.1 Firebase Setup

Firstly, Firebase needs to be connected to the Android Studio project. Then multiple SDKs need to be installed, they can be installed by selecting Firebase within the tool section of Android Studio. The following SDKs will be used within this project.

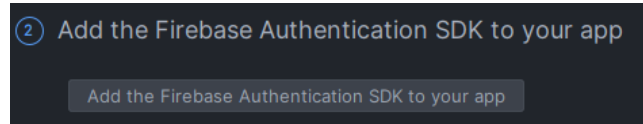


Figure 23: Authentication SDK

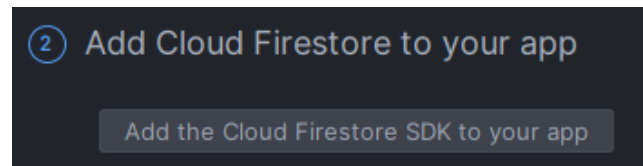


Figure 24: Firestore SDK

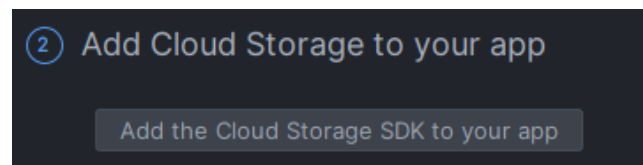


Figure 25: Storage SDK

It is important to import all Firebase related packages, but this is done whilst developing the application as Android Studio enables light packages. Every class must fetch the current user and storing value in a variable named user. This is critical, otherwise the application will not function accordingly.

```
FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
```

Figure 26: Current User Authentication

4.4.2 Main Activity

This is the main class, meaning that this class runs when the application is ran. The design developed previously within this report shows numerous landing pages which benefits the user by providing key information and engaging images, in return for a likely increase towards the user retention statistic. These landing pages are tailored towards first time users, meaning users that are running the application for their first time. This can be done by fetching shared preferences under the variable “prefs”. Then validated by the use of a validation statement where if it is a user’s first time, show landing page, if not then continue.

```
prefs = getSharedPreferences( name: "com.mycompany.myAppName", MODE_PRIVATE);
```

Figure 27: Shared Preferences

```
if (prefs.getBoolean( key: "firstrun", defValue: true)) {  
    // Do first run stuff here then set 'firstrun' as false  
    startActivity(new Intent( packageContext: MainActivity.this, landing_page_one.class));  
    // using the following line to edit/commit prefs  
    prefs.edit().putBoolean("firstrun", false).apply();  
    finish();  
}
```

Figure 28: First Run

Within the same if statement, it is also important to check if a user is signed in or not. If signed in, then a user can continue to their account. If not, then the user must be redirected to the login page.

```
FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
```

Figure 29: Current User

```
} else {  
    if (user != null) {  
        // User is signed in  
    } else {  
        startActivity(new Intent( packageContext: MainActivity.this, landing_page_logon.class)  
            overridePendingTransition(R.anim.slide_in_right, R.anim.slide_out_left);  
        finish();  
        // No user is signed in  
    }  
}
```

Figure 30: Check Login State

All transitions between pages are the same depending on if the user is going forward or backwards, all are a slide in transition at 300ms. This prevents any risk of epilepsy and creates a smooth experience to the user.

Kryspin Marcysiak

If it is the user's first run:



Figure 31:
Landing 1



Figure 32:
Landing 2

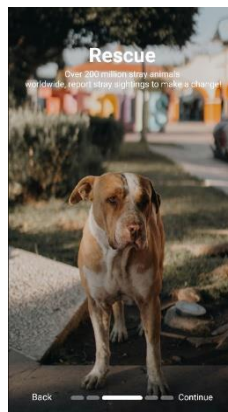


Figure 33:
Landing 3

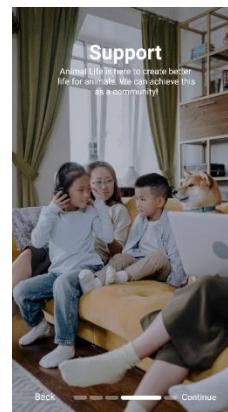


Figure 34:
Landing 4



Figure 35:
Landing 5

Once user is logged in, the app will display the homepage. This homepage is light and easily understandable. A top section displaying a hamburger menu, app title and account icon. The bottom navigation menu displaying the four main features on a low contrast background. The main content of the application is in a scroll view container.

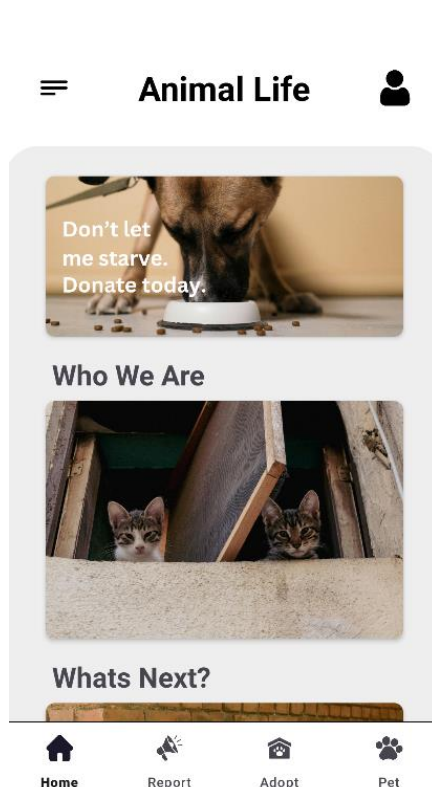


Figure 36: Homepage

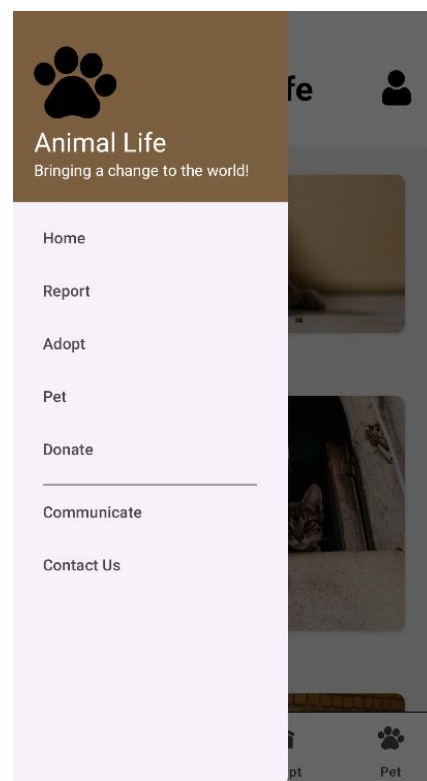


Figure 37: Navigation Menu

Kryspin Marcysiak

Animal Life

4.4.3 Sign In & Sign Up

To accomplish logon functionality, Firebase Authentication was used. There are three pages related to this functionality. But before they are functional, the java classes must be developed.

To create an account, values must be stored in local variables and validated by if statements that require the user to input a password longer than six characters. Once all data is imputed, using premade code provided by Firebase within the tool section mentioned previously and a few suitable changes, a user will be signed up and redirected to the sign in page to validate login.

```
public void onClick(View v) {
    String name = nameEditText.getText().toString().trim();
    String email = emailEditText.getText().toString().trim();
    String pass = passwordEditText.getText().toString().trim();

    if (email.isEmpty()) {
        emailEditText.setError("Email cannot be empty!");
    }
    if (pass.isEmpty()) {
        passwordEditText.setError("Password cannot be empty!");
    }
    if (pass.length() < 6) {
        passwordEditText.setError("Password must be longer than 6 characters!");
    }
    if (pass.isEmpty()) {
        passwordEditText.setError("Password cannot be empty!");
    }
    else {
        auth.createUserWithEmailAndPassword(email, pass).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    Toast.makeText(context, sign_up.this, text: "Sign Up Successful", Toast.LENGTH_SHORT).show();

                    userID = auth.getCurrentUser().getUid();
                    DocumentReference documentReference = firestore.collection(collectionPath: "users").document(userID);

                    Map<String, Object> user = new HashMap<>();
                    user.put("user_name", name);
                    user.put("user_email", email);

                    documentReference.set(user).addOnSuccessListener(new OnSuccessListener<Void>() {
                        @Override
                        public void onSuccess(Void unused) {
                            Toast.makeText(context, sign_up.this, text: "Welcome " + name, Toast.LENGTH_SHORT).show();
                        }
                    });
                }
            }
        });

        startActivity(new Intent(context, sign_in.class));
    }
}
```

Figure 38: Sign Up Class

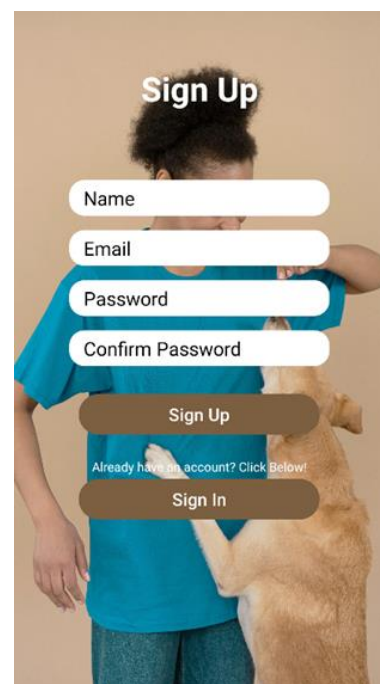


Figure 39: Sign Up Page

Once a user is redirected to the sign in page, using pre-made code provided by firebase tools, the application can authenticate a user into the application.

```
loginButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String email = loginEmail.getText().toString();
        String pass = loginPassword.getText().toString();

        if (!email.isEmpty() && Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
            if (!pass.isEmpty()) {
                auth.signInWithEmailAndPassword(email, pass)
                    .addOnSuccessListener(new OnSuccessListener<AuthResult>() {
                        @Override
                        public void onSuccess(AuthResult authResult) {
                            Toast.makeText(context, sign_in.this, "Login Successful", Toast.LENGTH_SHORT).show();
                            startActivity(new Intent(context, sign_in.this, MainActivity.class));
                            finish();
                        }
                    })
                    .addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                            Toast.makeText(context, sign_in.this, "Login Unsuccessful", Toast.LENGTH_SHORT).show();
                        }
                    });
            } else {
                loginPassword.setError("Password cannot be empty!");
            }
        } else if (email.isEmpty()) {
            loginEmail.setError("Email cannot be empty!");
        } else {
            loginEmail.setError("Please enter a valid email!");
        }
    }
});
```

Figure 40: Sign In Code

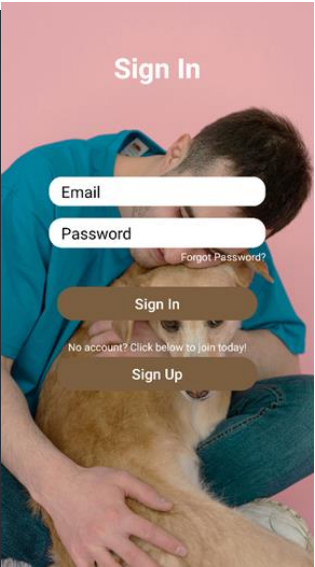


Figure 41: Sign In Page

To ensure the application works accordingly, a sign-up attempt is made using test data. Then when accessing the profile page, the inputted user information is displayed.

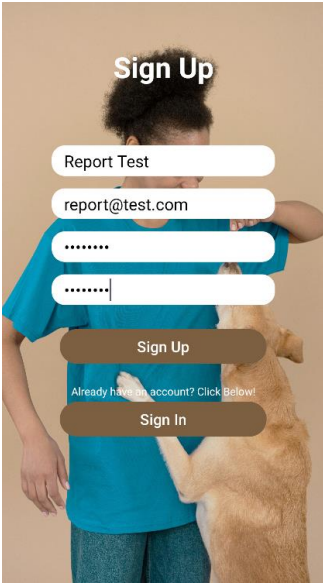


Figure 42: Sign Up Test

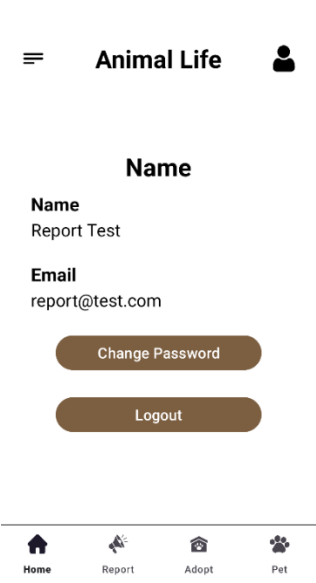


Figure 43: Account Confirmation

Resetting a password on an account is a must in the modern age. This can be done by Firebase's password reset template in the Authentication section.

```
change_password_text.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        EditText resetMail = new EditText(v.getContext());
        AlertDialog.Builder passwordResetDialog = new AlertDialog.Builder(v.getContext());
        passwordResetDialog.setTitle("Reset Password?");
        passwordResetDialog.setMessage("Enter your email address to receive a reset link.");
        passwordResetDialog.setView(resetMail);

        passwordResetDialog.setPositiveButton(text: "Yes", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                // Extract Email and send reset link

                String mail = resetMail.getText().toString();
                auth.sendPasswordResetEmail(mail).addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void unused) {
                        Toast.makeText(context, sign_in.this, text: "Reset link sent to email!", Toast.LENGTH_SHORT).show();
                    }
                }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(context, sign_in.this, text: "Error! Reset Link Not Sent!" + e.getMessage(), Toast.LENGTH_SHORT).show();
                    }
                });
            }
        });

        passwordResetDialog.setNegativeButton(text: "No", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                // Decline and close
            }
        });

        passwordResetDialog.create().show();
    }
});
```

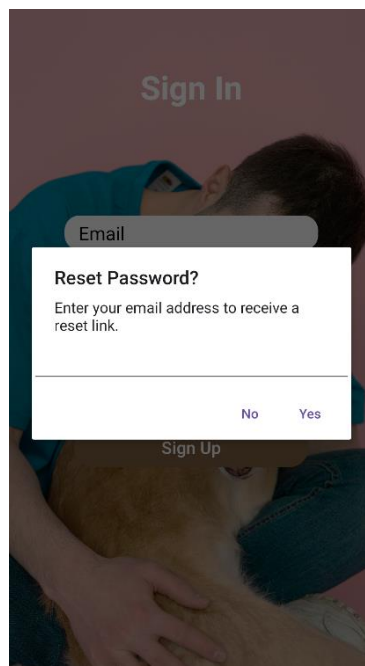


Figure 44: Password Reset Code

Figure 45: Password Reset Alert

When this request is submitted, an email in below form is sent to the recipient's email.

Kryspin Marcysiak

Subject
Reset your password for %APP_NAME%

Message

Hello,

Follow this link to reset your %APP_NAME% password for your %EMAIL% account.

https://animal-life-bd146.firebaseio.com/__/auth/action?mode=action&oobCode=code

If you didn't ask to reset your password, you can ignore this email.

Thanks,

Your %APP_NAME% team

Figure 46: Email Template

4.4.4 Report

The report feature of this mobile application require to gather form data inputted by the user to submit for an administrative review. An important aspect which should be considered when developing a form is to minimise user failure. This involves the use of spinners, these are drop down menus that will show a limited number of options, in this case only displaying a few types of likely animals to the user and an option saying “other” will reduce mistakes. This will not cause the user to be frustrated, will improve administrative experience, and increase efficiency.

The spinners enabling a user to select type of animal and its state must be declared for the application to be functional. This presents the user with limited options for the reason above.

```
// Initialize Spinners
spinnerState = findViewById(R.id.drop_down_animal_state);
spinnerType = findViewById(R.id.drop_down_animal_type);

// Define arrays of items for each Spinner
String[] itemsType = {"", "Dog", "Cat", "Horse", "Deer", "Other"};
String[] itemsState = {"", "Stray", "Distressed", "Hurt", "Missing", "Found", "Other"};

// Create ArrayAdapter for each Spinner using the respective array of items and a default spinner layout
ArrayAdapter<String> adapterState = new ArrayAdapter<>(context: this, android.R.layout.simple_spinner_item, itemsState);
ArrayAdapter<String> adapterType = new ArrayAdapter<>(context: this, android.R.layout.simple_spinner_item, itemsType);

// Specify the layout to use when the list of choices appears for each Spinner
adapterState.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
adapterType.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

// Apply the adapters to the Spinners
spinnerState.setAdapter(adapterState);
spinnerType.setAdapter(adapterType);
```

Figure 47: Report Spinners

As an image must be uploaded for the report to be submitted, relevant permissions must be gathered. This can be done by running a function that will begin a new external media intent.

```
private void openGallery() {
    Intent accessGallery = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    startActivityForResult(accessGallery, requestCode: 1);
}
```

Figure 48: Gallery Permission

To ensure all gathered data can be prepared to be uploaded into the reports collection as a document, data must be converted and passed. A random report ID will be the unique identification of the document being inserted; the image URL is to be converted into string for future access of it.

```
private String generateReportId() { return UUID.randomUUID().toString(); }

// Method to upload report
// Usage
private void uploadReport() {
    // Generate report ID
    reportId = generateReportId();

    // Get selected image URI
    if (imageUrl != null && spinnerType.getSelectedItemId() > 0 && spinnerState.getSelectedItemId() > 0 && !addressField.getText().toString().isEmpty() && !notesField.getText().toString().isEmpty()) {
        StorageReference imageRef = storageReference.child(pamDmgg: "images").child(pamDmgg: reportId + ".jpg");

        // Upload image to Firebase Storage
        imageRef.putFile(imageUri).addOnSuccessListener(taskSnapshot -> {
            // Image uploaded successfully
            // Get the download URL of the image
            imageRef.getDownloadUrl().addOnSuccessListener(uri -> {
                // Save the image URL
                imageUrl = uri.toString();

                // Upload the report data to Firebase Database
                uploadReportData();
            });
        }).addOnFailureListener(e -> {
            // Handle unsuccessful uploads
            Toast.makeText(context: report.this, text: "Failed to Upload Image", duration: Toast.LENGTH_SHORT).show();
        });
    } else {
        // No image selected
        Toast.makeText(context: report.this, text: "Form Incomplete!", duration: Toast.LENGTH_SHORT).show();
    }
}
```

Figure 49: Report Data Preparation

Once all data is ready to be uploaded into Firestore and Firebase Cloud Storage. A new document within the reports collection is locally declared. A new hash map is created, values are being converted into strings and inserted into the map. Then all data is uploaded into Firestore.

```
private void uploadReportData() {
    // Create a new document in the "reports" collection with the generated report ID
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    DocumentReference reportRef = db.collection(collectionPath: "reports").document(reportId);

    // Create a Map to hold the report data
    Map<String, Object> reportData = new HashMap<>();
    reportData.put("reportId", reportId);
    reportData.put("spinnerType", spinnerType.getSelectedItem().toString());
    reportData.put("spinnerState", spinnerState.getSelectedItem().toString());
    reportData.put("address", addressField.getText().toString());
    reportData.put("notes", notesField.getText().toString());
    reportData.put("imageUrl", imageUrl); // imageUrl obtained from uploading image

    // Add the report data to the Firestore document
    reportRef.set(reportData)
        .addOnSuccessListener(aVoid -> {
            // Report data uploaded successfully
            Toast.makeText(context: report.this, text: "Report Uploaded Successfully", duration: Toast.LENGTH_SHORT).show();
            startActivity(new Intent(packageContext: report.this, report_thank.class));
            overridePendingTransition(R.anim.slide_in_right, R.anim.slide_out_left);
        })
        .addOnFailureListener(e -> {
            // Handle errors
            Toast.makeText(context: report.this, text: "Failed to Upload Report", duration: Toast.LENGTH_SHORT).show();
        });
}
```

Figure 50: Report Data Insertion

Kryspin Marcysiak

To test the functionality of the above code, test data can be used to be inserted into the database.

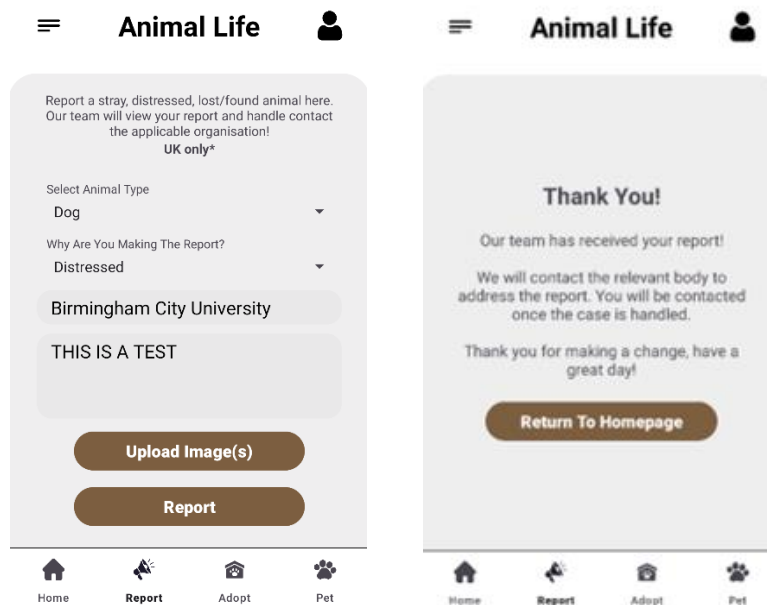


Figure 51: Report Page Test Figure 52: Report Confirmation

```
address: "Birmingham City University"
imageUrl: "https://firebasestorage.googleapis.com/v0/b/animal-life-bd146.appspot.com/o/images%2F38c9fec6-4813-4e59-94b4-744802e7fa74.jpg?alt=media&token=51eaf877-039c-4eca-a7b8-bca1a186e6f2"
notes: "THIS IS A TEST"
reportId: "38c9fec6-4813-4e59-94b4-744802e7fa74"
spinnerState: "Distressed"
spinnerType: "Dog"
```

Figure 53: Report Firestore Values

4.4.5 Adopt

The adoption feature within the mobile application will need to display all current adoption adverts from the Firestore database. Cooperating with the design, the use of clickable buttons to filter cats and dogs was challenging. Therefore, two horizontal scroll views within a scroll view present dogs and cats separate. Whilst the image is a dog, this is just for test purposes. Presenting the user with an image, a name, sex, breed, and age.

Fetching all advertisements within the collection where the pet type is equal to dog allows for only dog adverts to be displayed. By the support of query snapshots, values can be gathered from a document and set as variables.

```
Firestore.collection( collectionPath: "adopt-adverts").whereEqualTo( field: "pet-type", value: "Dog").get()
    .addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {
        @Override
        public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
            for (QueryDocumentSnapshot documentSnapshot : queryDocumentSnapshots) {
                // Get data from Firestore document
                String advertId = documentSnapshot.getId();
                String petName = documentSnapshot.getString( field: "pet-name");
                String petType = documentSnapshot.getString( field: "pet-type");
                String petBreed = documentSnapshot.getString( field: "pet-breed");
                String petImageUrl = documentSnapshot.getString( field: "pet-image");
                String petSex = documentSnapshot.getString( field: "pet-sex");
                String petDescription = documentSnapshot.getString( field: "pet-description");
                long petWeight = documentSnapshot.getLong( field: "pet-weight");
                boolean petVaccine = documentSnapshot.getBoolean( field: "pet-vaccine");
                Timestamp petBirthDate = documentSnapshot.getTimestamp( field: "pet-birthday");
                String petEmail = documentSnapshot.getString( field: "pet-email");
```

Figure 54: Fetching Adoption Data

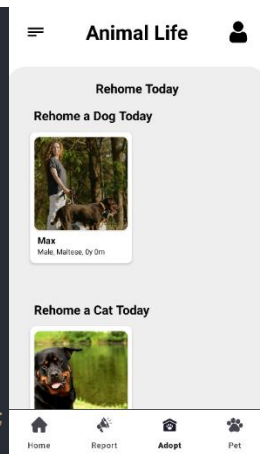


Figure 55: Adopt Page

Due to the timestamp used within Android Studio and Firebase being different, the timestamps must be calculated.

```
long currentTime = System.currentTimeMillis();
long birthTime = petBirthDate.toDate().getTime();
long ageInMillis = currentTime - birthTime;

//seconds to years and months
int years = (int) (ageInMillis / (1000 * 60 * 60 * 24 * 365.25));
int months = (int) ((ageInMillis % (1000 * 60 * 60 * 24 * 365.25)) / (1000 * 60 * 60 * 24 * 30.44));
```

Figure 56: Timestamp Calculation

More information needs to be displayed to the user about a pet that is up for adoption. This can be done by holding an empty layout and populating it with data relevant to the selected advertisement. Using intent and passing variables onto another class is the best way to do this.

```
cardView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent( packageContext: adopt.this, individual_adopt_page.class);
        // Pass relevant details to the new activity
        intent.putExtra( name: "advertId", advertId);
        intent.putExtra( name: "petName", petName);
        intent.putExtra( name: "petType", petType);
        intent.putExtra( name: "petBreed", petBreed);
        intent.putExtra( name: "petImageUrl", petImageUrl);
        intent.putExtra( name: "petSex", petSex);
        intent.putExtra( name: "petDescription", petDescription);
        intent.putExtra( name: "petWeight", petWeight);
        intent.putExtra( name: "petVaccine", petVaccine);
        intent.putExtra( name: "petAge", petAge);
        intent.putExtra( name: "petEmail", petEmail);
        // Add other fields as needed
        startActivity(intent);
    }
});

// Add the inflated CardView to your adsLayout
adsLayout.addView(cardView);
```

Figure 57: Individual Advert Intent

To populate an empty layout, each variable must be assigned to an id of an element. Once that is done, it is time to populate the interface with relevant data.

```
nameOfPet.setText(petName);
breedOfPet.setText(petBreed);
genderOfPet.setText(petGender);
weightOfPet.setText(petWeight + "kg");
descriptionOfPet.setText(petDescription);
vaccineOfPet.setText(petVaccine ? "Yes" : "No");
ageOfPet.setText(petAge);
```

All adoption requests are to be sent by email. To create a lighter experience for the user, this can be done by pre-setting email content.

Kryspin Marcysiak

```

toEmail = petEmail;
emailSubject = "Adoption of " + petName;
emailBody = "Hi, \n\nI am interested in the adoption of " + petName + ". Looking forward for your response.\n\nKind Regards,\n";

```

Figure 58: Email Content

It is important that each adoption request is also stored in Firestore and has the user's ID associated for reference and bookkeeping purposes to prevent Gmail client sided issues.

```

requestId = generateRequestId();
FirebaseFirestore db = FirebaseFirestore.getInstance();
DocumentReference reportRef = db.collection( collectionPath: "adoption-requests").document(requestId);

Map<String, Object> requestData = new HashMap<>();
requestData.put("adoption-request-id", requestId);
requestData.put("user-id", userId);

reportRef.set(requestData)
    .addOnSuccessListener(aVoid -> {
        Toast.makeText( context: individual_adopt_page.this, text: "Report Uploaded Successfully", Toast.LENGTH_SHORT).show();
        sendEmail();
    })

```

Figure 59: Adoption Request Firestore

To send an email containing preset content, a new intent must be started. Using the rfc822 setting, the application will open a share function on the mobile phone, prompting the user to select an option or back out.

```

private void sendEmail() {
    Intent intent = new Intent(Intent.ACTION_SEND);
    intent.putExtra(Intent.EXTRA_EMAIL, new String[]{toEmail});
    intent.putExtra(Intent.EXTRA_SUBJECT, emailSubject);
    intent.putExtra(Intent.EXTRA_TEXT, emailBody);
    intent.setType("message/rfc822");

    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
        Snackbar.make(findViewById(android.R.id.content), text: "We will contact you within 48 hours!", Snackbar.LENGTH_LONG).show();
    } else {
        Toast.makeText( context: individual_adopt_page.this, text: "There is no application that supports this action!", Toast.LENGTH_SHORT).show();
    }
}

```

Figure 60: Send Adoption Request

Below is the output of the selected advert, once the adopt button is pressed, the user is prompted to select an email provider. The preset email content auto fills to fill user satisfaction.

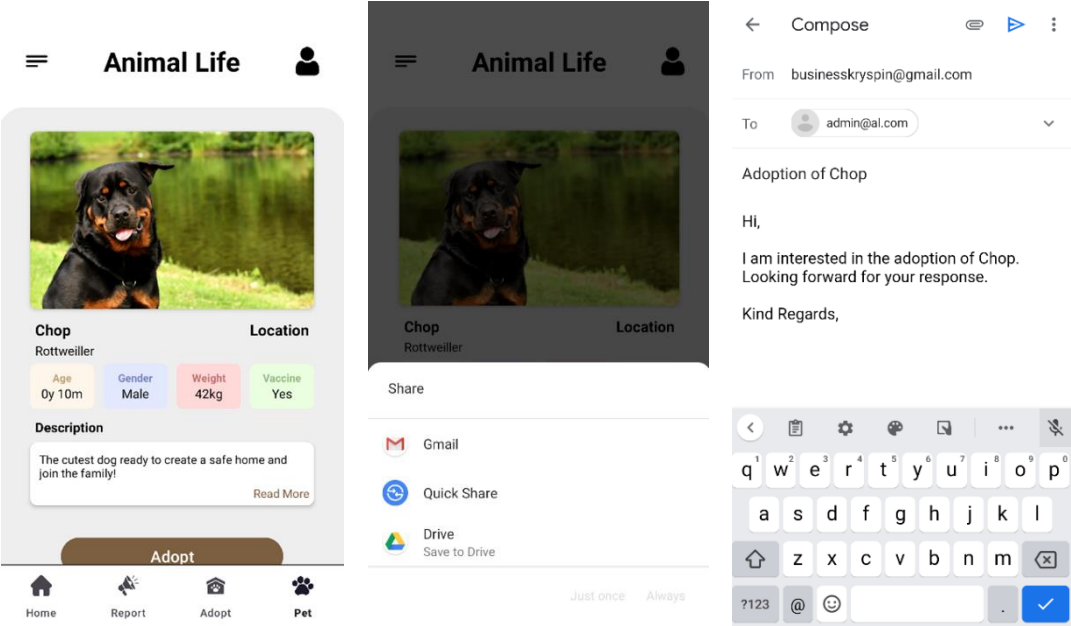


Figure 61: Adoption Pet Page Figure 62: Email Option Figure 63: Preset Email Content

The pet section is inactive due to Android Studio based issues. As of 05/05/2024.

4.5 Web Application

To develop this web application, it is important to host the application locally. The adequate choice of compiler for this project is Visual Studio due to its versatility and its “LiveServer” extension which will host the application locally. Using HTML, CSS, and JavaScript to develop this application will ensure for modernization.

4.5.1 Setting Up Connection

To create a connection between a web application and Firebase, generating unique configurational code is necessary. Within project settings, selecting “web” and naming the application, Firebase will generate code with unique connection credentials. I blurred by credentials for security purposes.

Then, initialize Firebase and begin using the SDKs for the products you'd like to use.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSvB88F_WAPnOG6Ars0F90B87-Is4n0wmgVg",
  authDomain: [REDACTED],
  projectId: [REDACTED],
  storageBucket: [REDACTED],
  messagingSenderId: [REDACTED],
  appId: [REDACTED],
  measurementId: "G-Q9mbs-mk11"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

Figure 64: Configuration Values

It is important this is in every document associated with Firebase to ensure a connection is present. In this case I will be using “var” instead of “const” as I was facing unknown issues.

```
var firebaseConfig = {
  apiKey: "AIzaSyCUVeWvvU57k_BiQ9H0wmS8etKbDDZbugE",
  authDomain: [REDACTED],
  databaseURL: [REDACTED],
  projectId: "ar",
  storageBucket: [REDACTED],
  messagingSenderId: [REDACTED],
  appId: "1:1",
  measurementId: "G-KZ5BE1SPZL"
};
firebase.initializeApp(firebaseConfig);
```

Figure 65: HTML Connection

There is no way to ensure a connection is present until actions such as login are made.
Kryspin Marcysiak

4.5.2 Base Head Content

Using standardised meta content as shown below, ensure that the web application is universally accessible for any screen size.

```
<meta charset="UTF-8">
<meta name="description" content="Animal Life">
<meta name="keywords" content="TODO">
<meta name="author" content="Kryspin Marcysiak">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Animal Life</title>
```

Figure 66: HTML Head Content

4.5.3 Logon Page

When taking the design of the logon page, it is clear that a form only accepting log-in data is required. Using “form” and multiple “div” tags for CSS styling, the main form container can be created. Within the form, two input boxes accompanied by labels and placeholders indicating each labels purpose. Finally, two script tags are used to reinforce the connection between Firebase and the app.

```
<div class="title">
  <h1>Animal Life</h1>
</div>

<form action="#" method="post">
  <div class="login_form">
    <label for="email"><b>Email</b></label><br>
    <input id="email-field" type="email" placeholder="email@domain.com" name="email" required><br><br>
    <label for="password"><b>Password</b></label><br>
    <input id="password-field" type="password" placeholder="*****" name="password" required><br>
    <p>Experiencing an issue? <a href="#">Contact Us</a></p>
    <div class="button_div">
      <button id="login-button" type="button" class="login_button">Login</button>
    </div>
  </div>
</form>

<p class="network">Interested in working with us? <a href="#">Contact Us</a></p>

<script src="https://www.gstatic.com/firebasejs/6.3.3/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/6.3.3/firebase-auth.js"></script>
```

Figure 67: HTML Logon Body

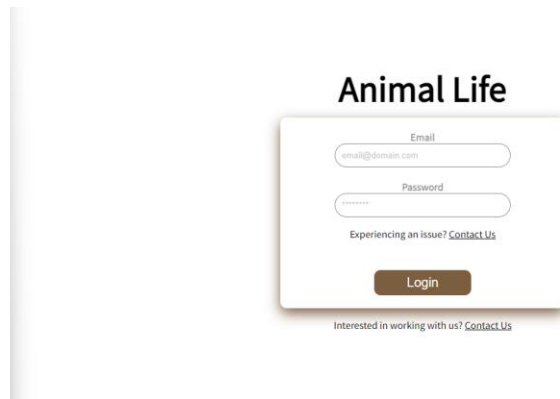


Figure 68: HTML Logon Output

Through the use of HTML and CSS code, the form has been created. But for this form to be functional, a java script has to be implemented. Below the setup code of firebase, a variable “auth” has to be declared for the firebase authentication feature to work accordingly.

```
firebase.initializeApp(firebaseConfig);
var auth = firebase.auth();
```

Figure 69: JavaScript Firebase Initialization

It is important to ensure that the form accesses Firebase only once the login button is pressed, this can be done by the use of an event listener. This even listener will store inputted values as variables, an if statement will check if the credentials match. If credentials match then the user will be granted access and notified, if not then the user will be notified of login failure.

```
document.getElementById("login-button").addEventListener("click", function (event) {
    event.preventDefault();

    var email = document.getElementById("email-field").value;
    var password = document.getElementById("password-field").value;

    auth.signInWithEmailAndPassword(email, password)
        .then(function (userCredential) {
            var user = userCredential.user;
            if (email === "admin@al.com" && password === "Password") {
                alert("Login successful!");
                window.location.href = "landing.html";
            } else {
                auth.signOut().then(function () {
                    alert("You do not have access to this portal <3");
                }).catch(function (error) {
                    console.error("Error signing out:", error);
                });
            }
        })
        .catch(function (error) {
            var errorCode = error.code;
            var errorMessage = error.message;
            alert("Please enter an email and password");
        });
});
```

Figure 70: JavaScript Login Event

It is important that the user remains logged in and not just directed onto the next page. This can be done by the use of local persistence.

```
auth.setPersistence(firebase.auth.Auth.Persistence.LOCAL)
  .then(function () {
  })
  .catch(function (error) {
    console.error("Error setting persistence:", error);
  });
```

Figure 71: JavaScript Remain Logged In

To provide a confirmation of a successful login or failure, alerts are useful as they create a popup window with a message.

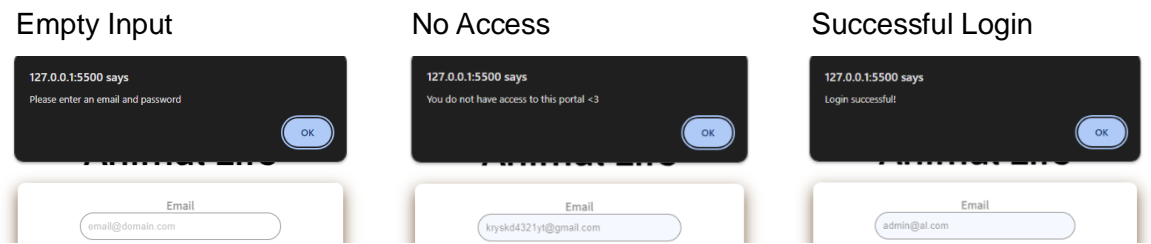


Figure 72: JavaScript Invalid Input

Figure 73 : JavaScript No Access

Figure 74: JavaScript Login Successful

Table 2: HTML Login Results

4.5.4 Side Navigation Menu

A simple side navigation menu which presents the user with four five options (dashboard, reports, adopt, educational and logout). The navigation menu must use the colour palette of the application to stay consistent.

```
<div id="sidebar">
  <h1>Animal Life</h1>
  <div class="side-menu">
    <ul>
      <li class="active"><a href="landing.html">Dashboard</a></li>
      <li><a href="reports.html">Reports</a></li>
      <li><a href="adopt.html">Adopt</a></li>
      <li><a href="educational.html">Educational</a></li>
      <li><a href="index.html">Logout</a></li>
    </ul>
  </div>
</div>
```

Figure 75: HTML Navigation Menu

```
#sidebar {
  position: fixed;
  left: 0;
  top: 0;
  height: 100%;
  width: 220px;
  background-color: #483821;
  padding-top: 15px;
}

#sidebar h1 {
  margin-bottom: 20px;
  text-align: center;
  color: white;
}

#sidebar ul {
  list-style-type: none;
  padding: 0;
}

#sidebar .side-menu ul {
  list-style: none;
  padding: 0;
}
```

Figure 76: Menu CSS 1

```
#sidebar .side-menu ul li a {
  color: white;
  text-decoration: none;
  display: block;
}

#sidebar .side-menu ul li.active {
  background-color: #6A5133;
}

#sidebar ul li {
  padding: 10px;
  color: white;
  text-align: center;
  user-select: none;
}

#form-title {
  text-align: center;
  margin-bottom: 10px;
  font-size: 20px;
}

#sidebar ul li: hover {
  background-color: #6A5133;
}
```

Figure 77: Menu CSS 2

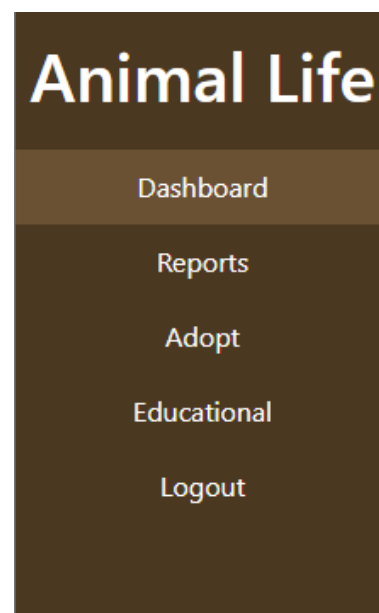


Figure 78: Menu Output

4.5.5 Dashboard Page

The dashboard page will be the main page of the web application, presenting analytics which are gathered by Google Analytics that is linked to Firebase automatically.

A Google Analytics tracking ID is required which can be found within the admin settings, data collection and modification, data streams and select the web application name “web-portal”.

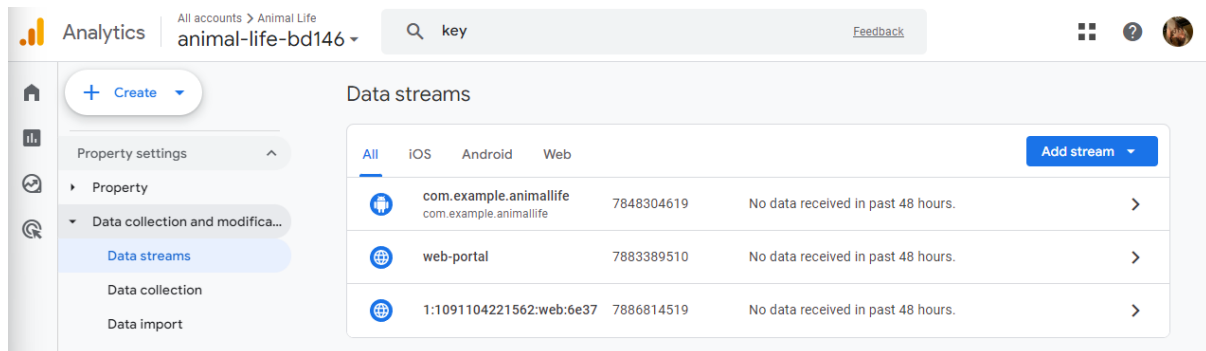


Figure 79: Google Analytics Setup

Select configure tab settings and copy the tag presented (I will not be showing this for security purposes).

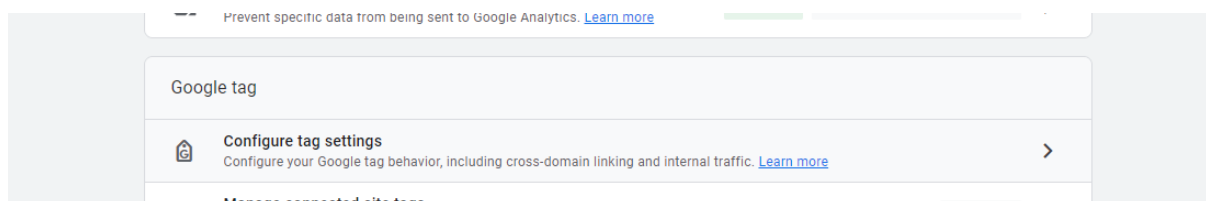


Figure 80: Google Analytics Tag

A link to the Google Analytics report is required which can be found within the report section.

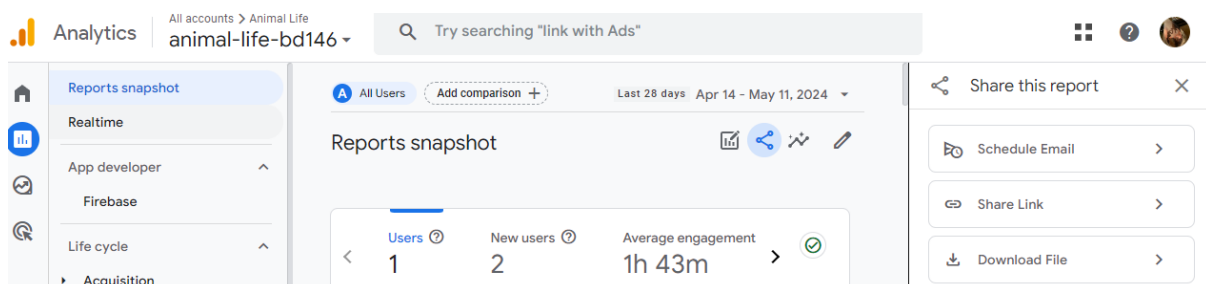


Figure 81: Google Analytics Report

The use of “iframe” can present all analytical data within a Google Analytics report.

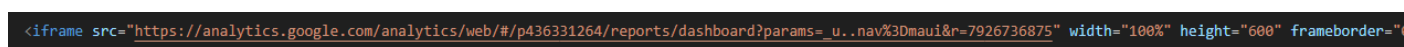


Figure 82: Google Analytics iframe

Kryspin Marcysiak

I have faced a major issue for the dashboard page as Google Analytics refused to connect. I was unable to find a solution to this problem, further research surrounding this issue to be done.

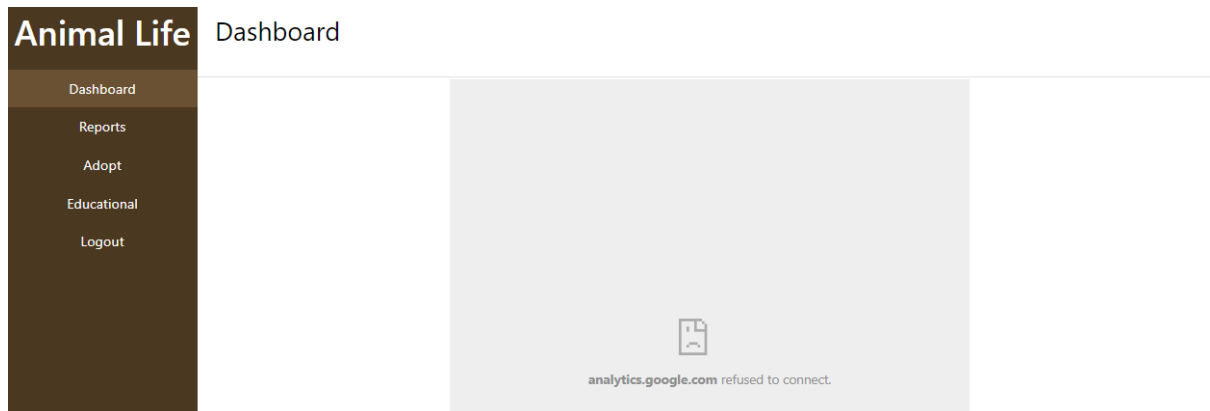


Figure 83: Dashboard Analytic Issue

4.5.6 Report Page

The most adequate way to present all reports to the user is by the use of a table. For this to be accomplished, a table must be created. In addition, a “tbody” tag will enable a future script to insert data. The table will use Bootstrap styling with the class “table table-hover”.

```
<table id="documentTable" class="table table-hover">

  <thead>
    <th>ID</th>
    <th>Pet Type</th>
    <th>Pet State</th>
    <th>Address</th>
    <th>Notes</th>
  </thead>
  <tbody id="tableBody"></tbody>
  <!-- Document data will be inserted here -->

</table>
```

Figure 84: HTML Report Table

The reports table will fetch data from a Firestore collection named “reports” and will present each document as an individual record. Firstly, variables must be imported from a firebase page and Firestore must be fetched.

```
import {
  doc, getDoc, getDocs, onSnapshot, collection, updateDoc
}
from "https://www.gstatic.com/firebasejs/10.11.0/firebase-firestore.js";

const db = getFirestore(app);
```

Figure 85: JavaScript Report Import

The reports collection must be fetched. By using query snapshot, all documents and its IDs from the collection can be fetched and passed onto another function.

```
async function getAllDataOnce() {
  const querySnapshot = await getDocs(collection(db, "reports"));
  var reports = [];

  querySnapshot.forEach(doc => {
    const reportsData = doc.data();
    reportsData.docId = doc.id;
    reports.push(reportsData);
  });

  AddAllItemsToTable(reports);
}
```

Figure 86: JavaScript Report Query

The following function uses passed document information and assigns its variables for further referencing.

```
function AddAllItemsToTable(AdoptAdvert) {
  no = 0;
  tableBody.innerHTML = "";
  AdoptAdvert.forEach(element => {
    AddItemToTable(element["spinnerType"], element["spinnerState"], element["address"], element["notes"], element["imageUrl"], element.docId);
  });
}
```

Figure 87: JavaScript Report Query

By the use of variables with individual created elements named “td”, the previous variables can be assigned. Finally appending all variables into the table rows, the table will present all found records within the database.

```
function AddItemToTable(spinnerType, spinnerState, address, notes, imageUrl, docId) {
  let tableRow = document.createElement("tr");
  let td1 = document.createElement("td");
  let td2 = document.createElement("td");
  let td3 = document.createElement("td");
  let td4 = document.createElement("td");
  let td5 = document.createElement("td");
  let td6 = document.createElement("td");
  let td7 = document.createElement("td");

  td1.innerHTML = ++no;
  td2.innerHTML = spinnerType;
  td3.innerHTML = spinnerState;
  td4.innerHTML = address;
  td5.innerHTML = notes;

  tableRow.appendChild(td1);
  tableRow.appendChild(td2);
  tableRow.appendChild(td3);
  tableRow.appendChild(td4);
  tableRow.appendChild(td5);
  tableRow.appendChild(td6);
  tableRow.appendChild(td7);

  tableBody.appendChild(tableRow);
}
```

Figure 88: JavaScript Report Content

Figure 89: JavaScript Report Rows

As records have been previously added via the mobile application, data should be visible.

Animal Life		Reports				
Dashboard		ID	Pet Type	Pet State	Address	Notes
Reports		1	Dog	Injured	29 Fullbrook Road, WS5 4PB	Right leg injured View Image
Adopt		2	Dog	Distressed	10 Riverwood Park, WS1 2PN	Constantly running away from people, no one can help! View Image
Educational						
Logout						

Figure 90: JavaScript Report Output

Presenting the administrator with pet type, its' state, location, and any notes from the user. They can now act accordingly.

4.5.7 Adoption Page

The adopt page will follow the same form setup as the previous page, but it will use different column titles and hold different data. The table will use Bootstrap styling with the class “table table-hover”.

```
<table id="documentTable" class="table table-hover">

  <thead>
    <th>ID</th>
    <th>Name</th>
    <th>Type</th>
    <th>Breed</th>
    <th>Sex</th>
    <th>Weight</th>
    <th>Vaccine</th>
  </thead>
  <tbody id="tableBody"></tbody>
  <!-- Document data will be inserted here -->

</table>
```

Figure 91: HTML Adoption Table

Another form which will enable a user to add an adoption advertisement must be added.

```
<form id="uploadForm">
  <button id="closeFormButton" class="closeButton">&times;</button>
  <div class="level">
    <label for="petName">Name</label>
    <input type="text" id="petName" name="petName" required>
  </div>
  <div class="level">
    <label for="petType">Type</label>
    <select id="petType" name="petType" required>
      <option value="Dog">Dog</option>
      <option value="Cat">Cat</option>
    </select>
  </div>
  <div class="level">
    <label for="petBreed">Breed</label>
    <input type="text" id="petBreed" name="petBreed" required>
  </div>
  <div class="level">
    <label for="petSex">Sex</label>
    <select id="petSex" name="petSex" required>
      <option value="Female">Female</option>
      <option value="Male">Male</option>
    </select>
  </div>
  <div class="level">
    <label for="petWeight">Weight (kg)</label>
    <input type="number" id="petWeight" name="petWeight" min="0" step="0.01" required>
  </div>
  <div class="level">
    <label for="petVaccine">Vaccine</label>
    <select id="petVaccine" name="petVaccine" required>
      <option value="Yes">Yes</option>
      <option value="No">No</option>
    </select>
  </div>
  <div class="level">
    <label for="petDescription">Description</label>
    <textarea id="petDescription" name="petDescription" required></textarea>
  </div>
  <div class="level">
    <label for="petBirthday">Birthday</label>
    <input type="datetime-local" id="petBirthday" name="petBirthday" required>
  </div>
  <div class="level">
    <label for="petImage">Image</label>
    <input type="file" id="petImage" name="petImage" accept="image/*" required>
  </div>
  <button type="submit">Submit</button>
</form>
```

Figure 92: HTML Adoption Popup 1

Figure 93: HTML Adoption Popup 2

A button which will make above form appear must be in the body of this HTML document.

```
<div class="topButtons">
  <button class="addAnimal" id="showFormButton">Add Animal for Adoption</button>
</div>
```

Figure 94: HTML Adoption Button

Firebase variables must be imported from all features that will be used. This includes features from firebase, Firestore and firebase storage. In addition, as modular code will be used, the script type must be set to “module”.

```
<script type="module">

import { initializeApp } from "https://www.gstatic.com/firebasejs/10.11.0/firebase-app.js";
import { getAnalytics } from "https://www.gstatic.com/firebasejs/10.11.0/firebase-analytics.js";
import { getFirestore, addDoc } from "https://www.gstatic.com/firebasejs/10.11.0/firebase-firestore.js";
import { getStorage, ref, uploadBytes, getDownloadURL } from "https://www.gstatic.com/firebasejs/10.11.0/firebase-storage.js";
```

Figure 95: JavaScript Adoption Import

The initialization of firebase, Firestore and firebase storage is required.

```
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
const storage = getStorage(app);
const db = getFirestore(app);
```

Figure 96: JavaScript Adoption Initialization

The reports collection must be fetched. By using query snapshot, all documents and its IDs from the collection can be fetched and passed onto another function.

```
async function getAllDataOnce() {
  const querySnapshot = await getDocs(collection(db, "adopt-adverts"));
  var adverts = [];

  querySnapshot.forEach(doc => {
    const advertData = doc.data();
    advertData.docId = doc.id;
    adverts.push(advertData);
  });

  AddAllItemsToTable(adverts);
}
```

Figure 97: JavaScript Adoption Document Fetch

A function uses passed document information and assigns its variables for further referencing within the script.

```
function AddAllItemsToTable(AdoptAdvert) {
  adoptNo = 0;
  tableBody.innerHTML = "";
  AdoptAdvert.forEach(element => {
    AddItemToTable(element["pet-name"], element["pet-type"], element["pet-breed"], element["pet-sex"],
  });
}
```

Figure 98: JavaScript Adoption Referencing 1

```
element["pet-weight"], element["pet-vaccine"], element["pet-description"], element["pet-birthday"], element["pet-image"], element.docId);
```

Figure 99: JavaScript Adoption Referencing 2

By the use of variables with individual created elements named “td”, the previous variables can be assigned accordingly. Finally appending all variables into the table rows, the table will present all records within the collection.

```
function AddItemToTable(petName, petType, petBreed, petSex, petWeight, petVaccine, petDescription, petBirthday, imageUrl, docId) {  
    let tableRow = document.createElement("tr");  
    let td1 = document.createElement("td");  
    let td2 = document.createElement("td");  
    let td3 = document.createElement("td");  
    let td4 = document.createElement("td");  
    let td5 = document.createElement("td");  
    let td6 = document.createElement("td");  
    let td7 = document.createElement("td");  
    let td8 = document.createElement("td");  
  
    td1.innerHTML = ++adoptNo;  
    td2.innerHTML = petName;  
    td3.innerHTML = petType;  
    td4.innerHTML = petBreed;  
    td5.innerHTML = petSex;  
    td6.innerHTML = petWeight;  
    td7.innerHTML = petVaccine;
```

Figure 100: JavaScript Adoption Row Fill 1

```
    tableRow.appendChild(td1);  
    tableRow.appendChild(td2);  
    tableRow.appendChild(td3);  
    tableRow.appendChild(td4);  
    tableRow.appendChild(td5);  
    tableRow.appendChild(td5);  
    tableRow.appendChild(td6);  
    tableRow.appendChild(td7);  
    tableRow.appendChild(td8);  
  
    tableBody.appendChild(tableRow);
```

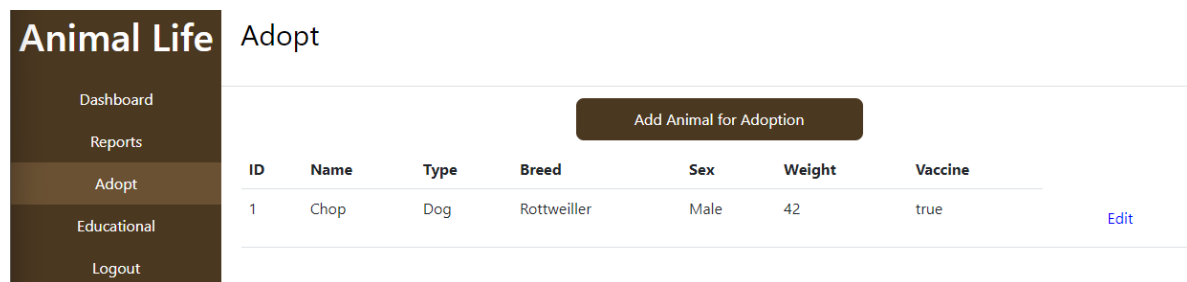
Figure 101: JavaScript Adoption Row Fill 2

The variable “td8” will be the eighth column, but instead of database data being displayed, an “edit” button will be present. It is important to apply styling properties to the button, passing the document id is also necessary to ensure that the correct information is displayed.

```
let editButton = document.createElement("button");  
editButton.textContent = "Edit";  
editButton.style.background = "none";  
editButton.style.border = "none";  
editButton.style.color = "blue";  
editButton.style.cursor = "pointer";  
editButton.setAttribute("data-doc-id", docId);
```

Figure 102: JavaScript Adoption Button

Records manually inputted within firebase are now be displayed.



ID	Name	Type	Breed	Sex	Weight	Vaccine	
1	Chop	Dog	Rottweiler	Male	42	true	Edit

Figure 103: JavaScript Adoption Output

An event listener which collects the document id associated with the row passes fetched information onto another function.

```
editButton.addEventListener("click", function () {  
  let currentDocId = this.getAttribute("data-doc-id");  
  handleEdit(petName, petType, petBreed, petSex, petWeight, petVaccine, petDescription, petBirthday, imageUrl, currentDocId);  
});
```

Figure 104: JavaScript Adoption Button

The handle edit function collects data and creates a new form with multiple divs for styling purposes. A large count of variables has to be used to ensure the user is presented with valid options, an example being that the pet type can be changed by the use of a drop-down menu displaying either “Dog” or “Cat”. Due to this section containing over 100 lines of code, the screenshots will be in [Appendix 2](#).

Within the edit function, an event listener for the submit button must be present. Data alike pet weight, vaccine and birthday have been converted from the firebase values to HTML values, but prior to updating the database, they must be converted back within the event listener.

```
let submitButton = document.createElement("button");  
submitButton.textContent = "Submit";  
submitButton.addEventListener("click", async function () {  
  const advertDocRef = doc(db, "adopt-adverts", docId);  
  console.log("docId:", docId);  
  
  let petWeightValue = parseFloat(weightInput.value);  
  
  let petVaccineValue = vaccineSelect.value === "true";  
  
  let petBirthdayValue = birthdayDate.getTime();
```

Figure 105: JavaScript Adoption Conversion

Finally, the “updateDoc” variable imported earlier can be used to update a document within a collection with new values.

```

try {
  await updateDoc(advertDocRef, {
    "pet-name": nameInput1.value,
    "pet-type": typeSelect.value,
    "pet-breed": breedInput.value,
    "pet-sex": sexSelect.value,
    "pet-weight": petWeightValue,
    "pet-vaccine": petVaccineValue,
    "pet-image": imageInput.src,
    "pet-description": descriptionInput.value,
    "pet-birthday": petBirthdayValue
  });
  console.log("Document updated successfully!");
  popup.remove();
} catch (error) {
  console.error("Error updating document: ", error);
}

```

Figure 106: JavaScript edit ad updateDoc

Upon selecting the edit text, a popup will appear with the selected documents data, displaying all values including the image.

The screenshot shows the 'Animal Life' web application interface. On the left is a sidebar with navigation links: Dashboard, Reports, Adopt, Educational, and Logout. The main content area is titled 'Adopt' and features a table with columns: ID, Name, Name, Breed, Sex, Weight, Vaccine, Pet Image, and Pet Birthday. The first row in the table shows a dog named 'Chop' with ID '1'. To the right of the table is an 'Edit' button. A modal form titled 'Add Animal for Adoption' is open, displaying the details for 'Chop'. The form includes input fields for Name (Chop), Type (Dog), Breed (Rottweiler), Sex (Male), Weight (42), Vaccine (Yes), and Pet Birthday (07/07/2023). There is a 'Pet Image' field showing a photo of a Rottweiler dog. Below these fields is a 'Pet Description' field with the text 'The cutest dog ready to create a safe home and join the family!'. A blue 'Submit' button is at the bottom of the form.

Figure 107: JavaScript Edit Ad

Once a value is changed and the submit button is pressed, the document does not update. Using the console provided within Google Chrome, I have found that there is the following error:

```

▲ [2024-05-12T00:19:50.356Z] @firebase/firestore:
Firestore (10.11.0): WebChannelConnection RPC 'Write'
stream 0x84551fbc transport errored:
  Qd {type: 'c', target: Q$1, g: Q$1, defaultPrevented:
    false, status: 1}

```

Figure 108: JavaScript Edit Ad Error

After attempting to troubleshoot this and no luck. I came across multiple articles speaking on this issue, it is clear that this is a common issue where Firebase is denying my edit request. As my read, write rules are adequate within Firebase and there seems to be no issue with my code, it is clear that this is a problem on Firebase's side.

Ongoing issue within a GitHub discussion can be found [here](#).

There is a button to add an adoption advert as well as a form that will only appear when the button is clicked. The use of functions to show and hide the form can do just that.

```
function showFormOverlay() {  
  document.getElementById('formOverlay').style.display = 'block';  
}  
  
function hideFormOverlay() {  
  document.getElementById('formOverlay').style.display = 'none';  
}
```

Figure 109: JavaScript Adoption Overlay

Figure 110: JavaScript Popup Form

Now that the form is visible, the fields must be stored in local variables, added onto a document, and stored within the adopt-adverts collection. A message confirming the name of the animal up for adoption ensure that the form worked as expected.

Kryspin Marcysiak

```

async function handleFormSubmit(event) {
  event.preventDefault();

  const petName = document.getElementById('petName').value;
  const petType = document.getElementById('petType').value;
  const petBreed = document.getElementById('petBreed').value;
  const petSex = document.getElementById('petSex').value;
  const petWeight = parseFloat(document.getElementById('petWeight').value);
  const petVaccine = document.getElementById('petVaccine').value === 'Yes';
  const petDescription = document.getElementById('petDescription').value;
  const petBirthday = new Date(document.getElementById('petBirthday').value);
  const petImageFile = document.getElementById('petImage').files[0];

  const imageRef = ref(storage, `petImages/${petImageFile.name}`);
  await uploadBytes(imageRef, petImageFile);

  const imageUrl = await getDownloadURL(imageRef);

  try {
    await addDoc(collection(db, 'adopt-adverts'), {
      "pet-name": petName,
      "pet-type": petType,
      "pet-breed": petBreed,
      "pet-sex": petSex,
      "pet-weight": petWeight,
      "pet-vaccine": petVaccine,
      "pet-description": petDescription,
      "pet-birthday": petBirthday,
      "pet-image": imageUrl
    });
    console.log('Document successfully written to Firestore!');
    alert("You have successfully added " + petName + " for adoption!");
    document.getElementById('uploadForm').reset();
  } catch (error) {
    console.error('Error writing document: ', error);
  }
}

```

Figure 111: JavaScript Popup Code

The following values will be inputted into the form and if successful, the table should be updated with an additional row.

ID	Name	Type	Breed	Sex	Weight (kg)	Vaccine	Description	Birthday	Image
1	Max	Dog	Maltese	Male	3.2	Yes	The cutest dog ever!	03/05/2024 01:35	happy_dog.jpg

Figure 112: JavaScript Popup Test

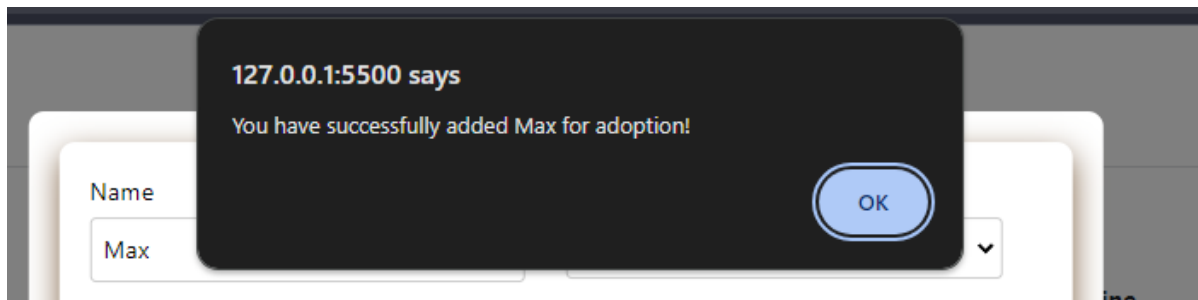


Figure 113: JavaScript Popup Confirmation

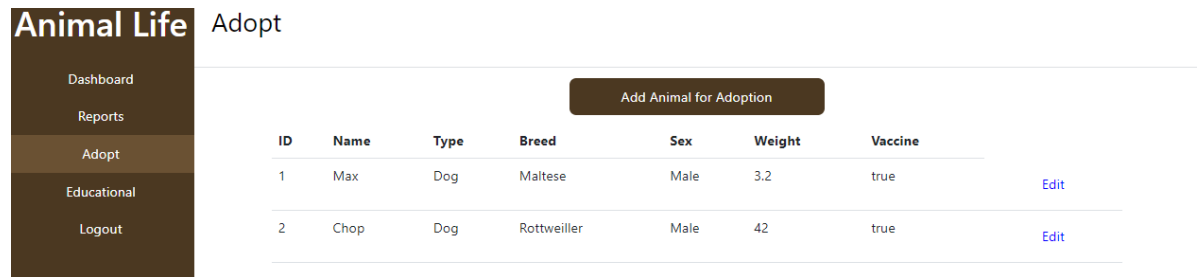


Figure 114: JavaScript Added Successfully

Once the form was submitted, a message confirming what pet was added has appeared. Then when refreshing the page, the new record is active.

4.5.8 Educational Page

Following the same process as when creating the adoption page but making sustainable changes, the table displays the following.

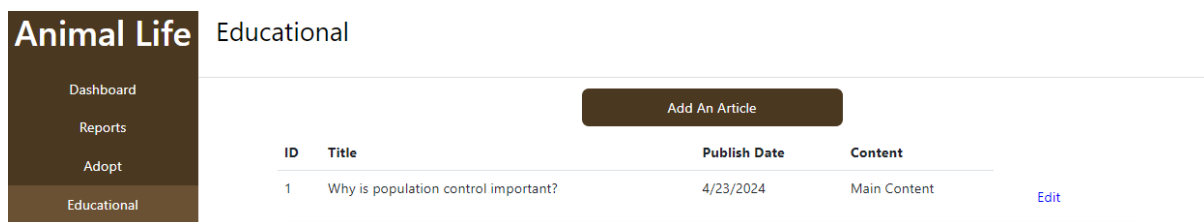


Figure 115: Educational Row

When the edit button is pressed, more information about the educational material is displayed but the same issue occurs.

Title:
Why is population control important?

Subtitle:
bla bla

Content:
Main Content

Publish Date:
23/04/2024

Image URL:
https://firebasestorage.googleapis.com/v0/b/animal-life-bd146.appspot.com/o/advert-ima

Submit

Figure 116: Educational Popup

```

[2024-05-12T00:48:01.848Z] logger.ts:209
@firebase/firestore: Firestore (10.11.0):
WebChannelConnection RPC 'Write' stream 0x5c3ad99d
transport errored:
  Qd {type: 'c', target: Q$1, g: Q$1, defaultPrevented:
    false, status: 1}

```

Figure 117: Educational Error

Upon adding an article with “test” values, the table has successfully been updated which ensures all works accordingly.

Animal Life		Educational		
Dashboard Reports Adopt Educational	ID	Title	Publish Date	Content
	1	Test Title	5/14/2024	Test
				Edit
	2	Why is population control important?	4/23/2024	Main Content
				Edit

Figure 118: Educational Added Successfully

5 Evaluation

A crucial part of developing any system is testing of the application. If applications are not tested, it may result in further issues and security vulnerabilities. The functionality, performance and user satisfaction are the most common choices.

5.1 Functionality Testing

Prior to conducting in person testing, the application must be functional with no errors. An adequate testing methodology to test the functionality of the mobile application is by using unit testing. Due to the current limitations presented by my system, Android Studio failed to load these tests.

Therefore, manual testing will be concluded on the reporting system and the adoption system. Both of the following sections will cover submission to viewing of report.

5.1.1 Report

Using test data, manual inputting the values in the table must be inputted into the application.


Animal Type	Animal State	Address	Notes	Image
Dog	Injured	10 Test Road	This is a test!	 <i>Figure 119: Report Test Image</i>

Table 3: Test Report Data

Once the report is made, we must check if the report has been added into the Firestore database.

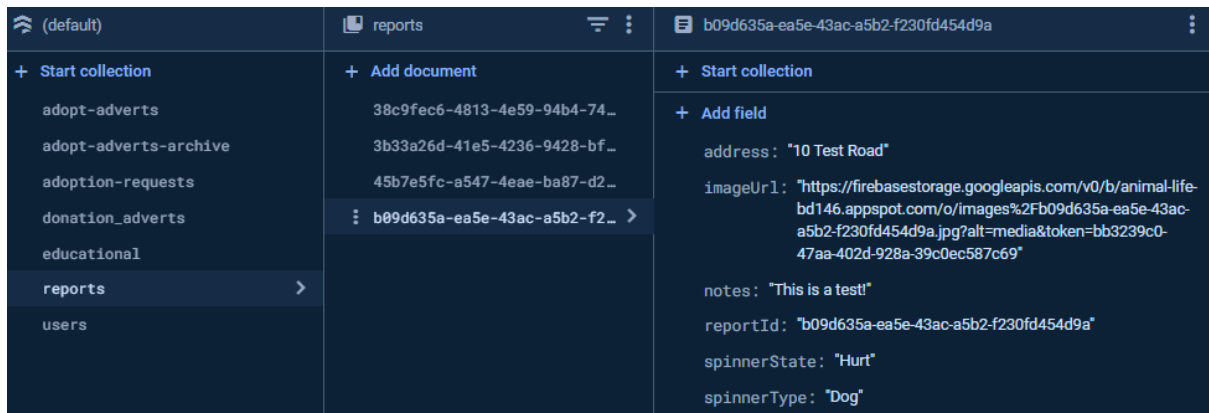


Figure 120: Report Database Update

Since the mobile application has successfully added the document into the reports collection, we must validate whether the web application reads the new added report.

4	Dog	Hurt	10 Test Road	This is a test!	View Image
---	-----	------	--------------	-----------------	----------------------------

Figure 121: Record added into table

Test has been successful.

5.1.2 Adoption Advert

To ensure the adoption feature works accordingly, test data found below must be entered into the add adoption advert form.


Name	Type	Breed	Sex	Vaccine	Weight	Birthday	Pet Image
Max	Dog	Gurry	Male	Yes	4.1	01/12/2002	 <i>Figure 122: Adoption Test Image</i>

Table 4: Adoption Test Data

Once the adoption advert has been uploaded, Firestore should display the new document.

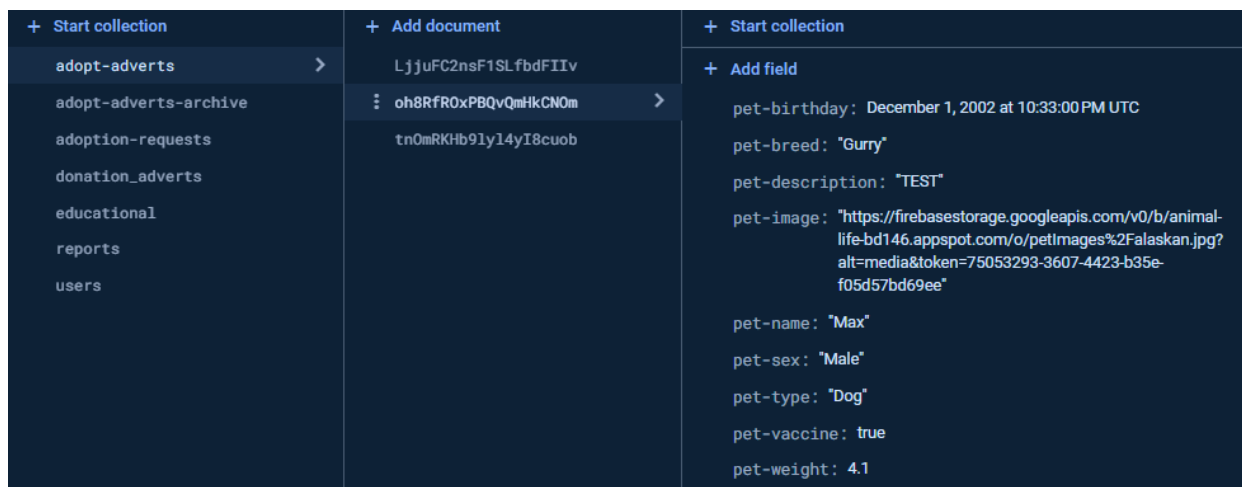


Figure 123: Adoption Added Into Firestore

The data has been successfully added, it is important to ensure the mobile application displays the advert within the dog section of the adopt page. In addition, once the advert is clicked, a page with relevant information should appear.

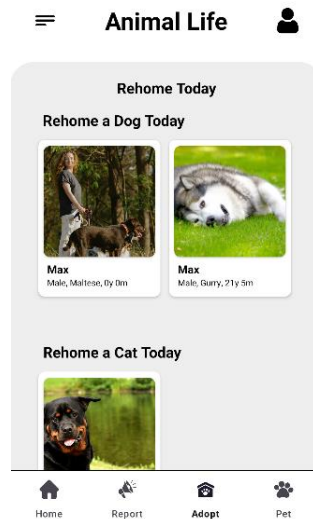


Figure 124: Adoption Page

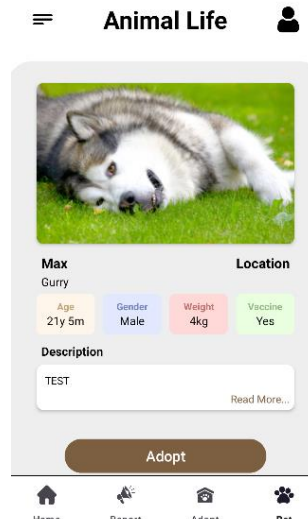


Figure 125: Unique Adoption Page

The adoption is successfully visible to the user, to ensure an adoption request can be submitted, the adopt button must be pressed to send an email. The email should then be received on the chosen platform, in this case Gmail.

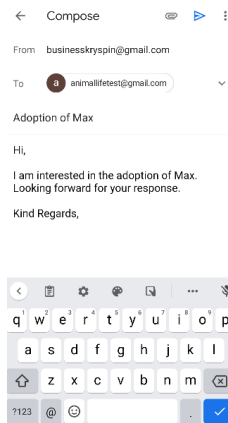


Figure 126: Email Test

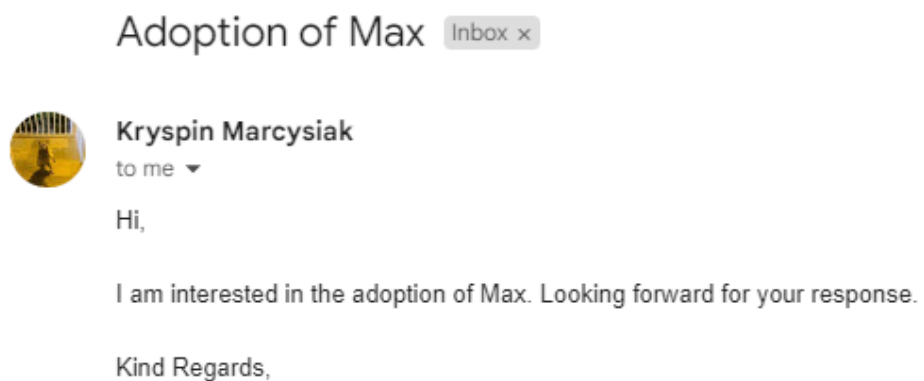


Figure 127: Email Arrival

Test has been successful.

Manual testing has confirmed that the main features of this system are fully functional with no issues.

Kryspin Marcysiak

5.2 Application Feedback

5.2.1 Feedback Criteria

Gathering feedback from people on the mobile application is essential as it can provide an average score and text-based feedback. Setting criteria for this is required as it sets the foundational questions which will be asked. The criteria for in person testing:

- How easy is it to operate this application?
- How easy is it to make a report?
- Do you like the layout and colour of the app?
- Do you have any suggestions?

5.2.2 Feedback Results

Metric	Average Score
Q1 – How easy is it to operate this application?	9
Q2 – How easy is it to make a report?	6.6
Q3 – Do you like the layout and colour of the app?	7.6

Table 5 User Testing

Questions four got short responses, two stating that when making a report, it should either use your location to make the report or use an address finder to be more precise. One other response stated that the application was too bright.

From user-based testing, it is clear that a location-based reporting feature should use either the current location or a precise address inputter. There should be a feature allowing a user to do both. This is because if a user is near an animal and want to make a report then they can use the current location. But if the user is making the report hours after, they will want to input a location familiar to the situation.

Kryspin Marcysiak

6 Conclusion

In conclusion, Animal Life is a step in the right direction towards higher quality animal welfare. Through the use of a mobile application on a widely accessible platform. Multiple features have been developed. A sighting report feature enabling users to provide any endangered animal an opportunity for a better life within animal shelters where there is food, care, and safety. An adoption feature which allows users to view animals in pet shelters, providing them a new warm-hearted family and freeing space within animal shelters for any other endangered animals.

Through developing this project from scratch and me developing a lot through trial and error, I have majorly developed my Java, Android Studio and JavaScript skills. Currently studying mobile development supported me with this module. While work from other modules, private life and mental health having an impact on the amount of time I was able to spend on this module, I tried my best and I am satisfied with the result.

Whilst I did face a wide range of issues such as the pet page within the mobile application stopping to work. The dashboard page within the web application not working accordingly. Also, whilst using a web application and attempting to update a document within a collection causes numerous issues due to Firebase client having ongoing issues. These are problems which I can learn from and improve on until Innovation Fest.

Kryspin Marcysiak

7 Recommendations For Future Work

If there is any future work that could be carried out in order to improve the Animal Life system, the following things can be done in order to improve it.

- A map on the web application showing reporting hotspots in order to determine the severity of endangered animals within an area. This would divert a higher capacity of rescue teams towards an area to reduce the likelihood of diseases spreading.
- An automated forwarding service that will automatically forward report cases to relevant bodies to reduce administration work. Whilst this has positives such as reducing workload, especially if there are a high volume of reports coming in. On the other hand, there is a change this might get out of hand if not done correctly.

8 Appendix

8.1 Appendix 1

4	21140376	Display Week:			30		29 Apr 2024					6 May 2024					13 May 2024										
5							29	30	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
6	TASK	PROGRESS			START	END	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S
8	Planning																										
9	Aim & Objectives & Search Methodology	100%	9/10/23	11/10/23																							
10	Research	100%	12/10/23	20/11/23																							
11	Literature Review & Themes	100%	22/11/23	2/12/23																							
12	Summary	100%	2/12/23	3/12/23																							
13	Design Methods & Development	100%	4/12/23	29/1/24																							
14	Deisgn																										
15	Low Fidelity Mobile Application	0%	30/1/24	31/1/24																							
16	Medium Fidelity Mobile Application	0%	1/2/24	4/2/24																							
17	High Fidelity Mobile Application	0%	5/2/24	8/2/24																							
18	User Testing of high Fidelity	0%	9/2/24	10/2/24																							
19	Improvements	0%	11/2/24	12/2/24																							
20	Low Fidelity Web Application	0%	11/2/24	12/2/24																							
21	Medium Fidelity Web Application	0%	13/2/24	18/2/24																							
22	High Fidelity Web Application	0%	19/2/24	22/2/24																							
23	User Testing of high Fidelity	0%	23/2/24	25/2/24																							
24	Improvements	0%	26/2/24	27/2/24																							
25	Development																										
26	Mobile Development	0%	28/2/24	8/4/24																							
27	Mobile Testing	0%	9/4/24	13/4/24																							
28	Web Development	0%	17/4/24	2/5/24																							
29	Web Testing	0%	3/5/24	7/5/24																							
30	Improvements	0%	2/5/24	6/5/24																							
31	Re-Evaluation																										
32	Evaluation	0%	9/5/24	12/5/24																							

Kryspin Marcysiak

8.2 Appendix 2

```
let popup = document.createElement("div");
popup.classList.add("popup");

let form = document.createElement("form");

let flexContainer1 = document.createElement("div");
flexContainer1.classList.add("flex-container");
let flexContainer2 = document.createElement("div");
flexContainer2.classList.add("flex-container");
let flexContainer3 = document.createElement("div");
flexContainer3.classList.add("flex-container");
let flexContainer4 = document.createElement("div");
flexContainer4.classList.add("flex-container");
let flexContainer5 = document.createElement("div");
flexContainer5.classList.add("flex-container");
let flexContainer6 = document.createElement("div");
flexContainer6.classList.add("flex-container");

let nameLabel1 = document.createElement("label");
nameLabel1.textContent = "Name:";
let nameInput1 = document.createElement("input");
nameInput1.value = petName;
let nameContainer = createContainerWithLabelAndInput(nameLabel1, nameInput1);
flexContainer1.appendChild(nameContainer);

let typeLabel = document.createElement("label");
typeLabel.textContent = "Type:";
let typeSelect = document.createElement("select");
typeSelect.classList.add("form-control");

let dogOption = document.createElement("option");
dogOption.value = "Dog";
dogOption.textContent = "Dog";

let catOption = document.createElement("option");
catOption.value = "Cat";
catOption.textContent = "Cat";

typeSelect.appendChild(dogOption);
typeSelect.appendChild(catOption);

typeSelect.value = petType;

let typeContainer = createContainerWithLabelAndInput(typeLabel, typeSelect);
flexContainer1.appendChild(typeContainer);

let breedLabel = document.createElement("label");
breedLabel.textContent = "Breed:";
let breedInput = document.createElement("input");
breedInput.value = petBreed;
let breedContainer = createContainerWithLabelAndInput(breedLabel, breedInput);
flexContainer2.appendChild(breedContainer);

let sexLabel = document.createElement("label");
sexLabel.textContent = "Sex:";
let sexSelect = document.createElement("select");
sexSelect.classList.add("form-control");

let maleOption = document.createElement("option");
maleOption.value = "Male";
maleOption.textContent = "Male";

let femaleOption = document.createElement("option");
femaleOption.value = "Female";
femaleOption.textContent = "Female";

sexSelect.appendChild(maleOption);
sexSelect.appendChild(femaleOption);

sexSelect.value = petSex;

let sexContainer = createContainerWithLabelAndInput(sexLabel, sexSelect);
flexContainer2.appendChild(sexContainer);

let weightLabel = document.createElement("label");
weightLabel.textContent = "Weight:";
let weightInput = document.createElement("input");
weightInput.value = petWeight;
let weightContainer = createContainerWithLabelAndInput(weightLabel, weightInput);
flexContainer3.appendChild(weightContainer);

let vaccineLabel = document.createElement("label");
vaccineLabel.textContent = "Vaccine:";
let vaccineSelect = document.createElement("select");
vaccineSelect.classList.add("form-control");

let yesOption = document.createElement("option");
yesOption.value = "true";
yesOption.textContent = "Yes";

let noOption = document.createElement("option");
noOption.value = "false";
noOption.textContent = "No";

vaccineSelect.appendChild(yesOption);
vaccineSelect.appendChild(noOption);

vaccineSelect.value = petVaccine;

let vaccineContainer = createContainerWithLabelAndInput(vaccineLabel, vaccineSelect);
flexContainer3.appendChild(vaccineContainer);

let imageLabel = document.createElement("label");
imageLabel.textContent = "Pet Image:";
let imageInput = document.createElement("img");
imageInput.src = imageUrl;
imageInput.style.maxWidth = "100%";
imageInput.style.borderRadius = "8px";
let imageContainer = createContainerWithLabelAndInput(imageLabel, imageInput);
flexContainer4.appendChild(imageContainer);

let birthdayLabel = document.createElement("label");
birthdayLabel.textContent = "Pet Birthday:";
let birthdayInput = document.createElement("input");
birthdayInput.type = "date";

let birthdayDate = new Date(petBirthday.seconds * 1000);
let year = birthdayDate.getFullYear();
let month = (birthdayDate.getMonth() + 1).toString().padStart(2, '0');
let day = birthdayDate.getDate().toString().padStart(2, '0');
birthdayInput.value = `${year}-${month}-${day}`;

let birthdayContainer = createContainerWithLabelAndInput(birthdayLabel, birthdayInput);
flexContainer5.appendChild(birthdayContainer);

let descriptionLabel = document.createElement("label");
descriptionLabel.textContent = "Pet Description:";
let descriptionInput = document.createElement("textarea");
descriptionInput.value = petDescription;
descriptionInput.style.height = "100px";

let descriptionContainer = createContainerWithLabelAndInput(descriptionLabel, descriptionInput);
flexContainer6.appendChild(descriptionContainer);
```

9 References

Cheryl.B (2023). What is a Landing Page & Why is it Important? [online] Available at:

<https://www.wsiworld.com/blog/the-importance-of-landing-pages>

Helen.F (2018). The secret science of successful landing pages [online] Available at:

<https://www.linkedin.com/pulse/secret-science-successful-landing-pages-helen-hammond/>

ESDAW (unknown). Stray animals by country - Europe [online]. Available at:

<https://www.esdaw.eu/stray-animals-by-country.html#:~:text=Stray%20animals%20by%20country%20%2D%20Europe&text=Number%20of%20abandoned%20and%20homeless%20dogs%20and%20cats%20in%20Europe,b%20e%20over%20100%20million%20animals>

ESDAW (unknown) The stray dogs in Europe [online]. Available from: [https://www.esdaw-](https://www.esdaw-eu.eu/the-stray-dogs-in-europe.html#:~:text=The%20European%20Union%20estimates%20that,of%20purebred%20and%20mixed%20breeds)

[eu.eu/the-stray-dogs-in-europe.html#:~:text=The%20European%20Union%20estimates%20that,of%20purebred%20and%20mixed%20breeds](https://www.esdaw-eu.eu/the-stray-dogs-in-europe.html#:~:text=The%20European%20Union%20estimates%20that,of%20purebred%20and%20mixed%20breeds)

Four-Paws (2023). STRAY ANIMALS: PETS WITHOUT A HOME [online]. Available at:

[https://www.four-paws.org/campaigns-topics/topics/help-for-stray-animals/stray-animals-pets-without-a-home#:~:text=The%20World%20Health%20Organisation%20\(WHO,the%20streets%20in%20different%20countries](https://www.four-paws.org/campaigns-topics/topics/help-for-stray-animals/stray-animals-pets-without-a-home#:~:text=The%20World%20Health%20Organisation%20(WHO,the%20streets%20in%20different%20countries)

Abubakar Abdulkarim, Mohd Azam Khan Bin Goriman Khan*, and Erkihun Aklilu (2021).

Stray Animal Population Control: Methods, Public Health Concern, Ethics, and Animal

Welfare Issues [online]. Available at: [https://eprints.science-](https://eprints.science-line.com/id/eprint/352/1/WVJ%2011%283%29%2C%20319-326%2C%20September%2025%2C%202021.pdf)

[line.com/id/eprint/352/1/WVJ%2011%283%29%2C%20319-326%2C%20September%2025%2C%202021.pdf](https://eprints.science-line.com/id/eprint/352/1/WVJ%2011%283%29%2C%20319-326%2C%20September%2025%2C%202021.pdf)

Lousia.T (2023). Stray Animal Control Practices (Europe) [online]. Available at:

https://www.strays.in/wp-content/uploads/2011/07/Stray-dog-control-practices_RSPCA.pdf

Kryspin Marcysiak

Svanhild.M (2022). Possible Solutions of the Stray Animal Problem [online]. Available at: <http://www.huveta.hu/bitstream/handle/10832/3552/513995337.pdf?sequence=1&isAllowed=y>

Kenny.T (2023). Americans can't afford their pets. It's pushing animal shelters to the brink [online]. Available at: <https://www.vox.com/future-perfect/2023/8/16/23833307/pets-animal-shelters-cats-dogs-affordable-housing-inflation>

Izzie.P (2019). Improving the Welfare of Companion Dogs—Is Owner Education the Solution? [online]. Available at: <https://www.mdpi.com/2076-2615/9/9/662>

Athanasios.S (2017). Mobile Applications within Education [online]. Available at: <https://online-journals.org/index.php/i-jim/article/view/6589/4406>

Mariam.A (unknown). User Experience Design (UXD) of Mobile Application: An Implementation of a Case Study [online]. Available at: <https://jtec.utem.edu.my/jtec/article/view/2902/2080>

Fakhri (2022). Improve your UI design with these successful UX Laws in 2023 [online]. Available at: <https://bootcamp.uxdesign.cc/improve-your-ui-with-these-successful-ux-laws-546d8ba027dc>

Xiangqian.F (2015). Mobile phone UI design principles in the design of human-machine interaction design [online]. Available at: <https://ieeexplore.ieee.org/document/5681254>

Xiaojun.B (2013). Fitts law: modelling finger touch with Fitts' law [online]. Available at: <https://dl.acm.org/doi/abs/10.1145/2470654.2466180>

Nasr.S (2014). User Interface Designing: Colour Therapy Sharing Application [online]. Available at: https://www.researchgate.net/profile/Mustafa-Saeed/publication/328150347_User_Interface_Designing_Colour_Therapy_Sharing_Application/links/5bbdb15299bf1049b75d9eb/User-Interface-Designing-Colour-Therapy-Sharing-Application.pdf

Kryspin Marcysiak

Lauren.P (2018). Expectations for dog ownership: Perceived physical, mental and psychosocial health consequences among prospective adopters [online]. Available at: <https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0200276&type=printable>

Goitom.T (2013). Does the Role a Pet Played before Disposition, and How the Pet is Lost Influence Pet Owner's Future Pet Adoption Decision? [online]. Available at: https://www.researchgate.net/profile/Goitom-Tesfom-tsegay/publication/253236529_Does_the_Role_a_Pet_Played_Before_Disposition_and_the_Way_the_Pet_is_Lost_Influence_Pet_Owner's_Future_Pet_Adoption_Decision/links/544310a30cf2a6a049a8a1f3/Does-the-Role-a-Pet-Played-Before-Disposition-and-the-Way-the-Pet-is-Lost-Influence-Pet-Owners-Future-Pet-Adoption-Decision.pdf?_sg%5B0%5D=started_experiment_milestone&origin=journalDetail

Amir.Z (2022). Pet analytics: Predicting adoption speed of pets from their online profiles [online]. Available at: <https://www.sciencedirect.com/science/article/pii/S0957417422009083>

Jeffery.H (2021). Did the COVID-19 Pandemic Spark a Public Interest in Pet Adoption? [online]. Available at: <https://www.frontiersin.org/articles/10.3389/fvets.2021.647308/full>

Odean.C (2013). Pets and Mental Health [online]. Available at: https://books.google.co.uk/books?hl=en&lr=&id=v0jJAwwAAQBAJ&oi=fnd&pg=PP1&dq=pets+and+mental+health&ots=fgX1Pq51bT&sig=aC6xfVv3Mb-H8CD5HGj0BUXqT64&redir_esc=y#v=onepage&q&f=false

Dasha.G (2021). The Effect of Pets on Human Mental Health and Wellbeing during COVID-19 Lockdown in Malaysia [online]. Available at: <https://www.mdpi.com/2076-2615/11/9/2689>

Stephen.L (2013). Potential Benefits of Canine Companionship for Military Veterans with Posttraumatic Stress Disorder (PTSD) [online]. Available at: https://brill.com/view/journals/soan/21/6/article-p568_4.xml

Kryspin Marcysiak

Terry.K (2018). How service dogs enhance a veterans occupational performance in the home: A qualitative perspective [online]. Available at:

<https://scholarworks.wmich.edu/cgi/viewcontent.cgi?article=1468&context=ojob>

Rochdale Borough Council (2022) Stray Dog Population[online]. Available from:

<https://data.europa.eu/data/datasets/stray-dog-information?locale=en>