School of Computer Science & Electrical Engineering

BSc Computer Science

Orion: Cat Scanner & Encyclopaedia - an iOS App implementing Machine Learning for Cat Detection and Breed Recognition

Final Year Project Report

Olaf Bogus
BOGUS03801

Supervised by: Professor John Gan
Second Accessor: Doctor Themistoklis Melissourgos

In memory of Orion.

The best cat there ever was.

You have provided me with endless inspiration and are the soul of my project.

Thank you, you are forever in my heart.

Rest in peace.

2005-2022

Acknowledgments

I would like to thank everyone that helped me and supported me through the creation of this project.

To every student, lecturer and staff member who has helped me during my degree.
My tutor Professor John Gan who gave me insightful advice every week and always made sure I was not panicking. Thank you for all your patience and help.
My partner Callum Roll for supporting me every step of the way and comforting me whenever the pressure got overwhelming.
My friend and fellow student Arti Durga for accompanying me through this degree, letting me vent all my frustrations and motivating me by studying and working with me.

I would like to give special thanks to my parents, who have helped me in many ways. Thank you for being so supportive and loving.
To my mom without whom I would not be in University of Essex thank you. This is all possible because of you.

Table of contents

Figures

Chapter 1:
Project Introduction

1.1 Project Choice Reasoning

To begin the report I would like to explain what motivated me to choose this project. While being presented with over 400 projects. I thought that it would be very hard to decide on just one but I was proved wrong.
While looking through the database I would add projects that I found interesting to a list and then go through it again eliminating less interesting projects until 5 were left. I was looking for a project that would not only be compelling to do but would also challenge me. Ideally it would involve something I was passionate about. The final year project had to be complex enough but would not overwhelming to the point of failure.
Throughout my degree I have really enjoyed Software Development modules and Machine Learning projects. The originally named “Developing A mobile app using machine learning for cat breed recognition” seemed perfect and spoke right to my heart. It had everything I wanted in a project. It allowed for a lot of flexibility. The complexity of the app could vary from very low to very advanced. The machine learning could have very low accuracy or thrive and be extremely accurate. I would have space in case of failure to achieve objectives as well as additional expansion if the core goals proved easy.
The key in the project were cats. I grew up as the only child with a cat being my sibling and my best friend. His name was Orion and that is where the app names comes from. I have had a passion for cats my whole life. In science felines are often overlooked in order to focus on canines - this holds true for Machine Learning as well. The availability of datasets for dogs and cats is vastly different. The number of classification projects focusing on dogs outnumbers cats greatly. This added an additional challenge to the project while researching something I had a deep passion for.

1.2 Project Goals

The aims of this project can be identified as two separate categories. The first one: to produce an iOS application that would perform three basic functions:

1. Accept a photo uploaded by the user either via their Photos library or Camera app then analyse it for cat breed features.
2. Save that photo and the information returned from a machine learning model and be capable of displaying the saved information
3. Present an encyclopaedia page consisting of cat breeds and their key information.

The second part was to do with machine learning models. The machine learning model would implement image classification. The classification would analyse the photo for cat breeds. The model would ideally achieve high accuracy and contain a large amount of labels, ideally above a dozen.
The project goals were split into Epics, Stories and Sub-Tasks using Jira.
The three main Epics that were identified are linked to the assessment milestones:

1. Challenge Week
2. Interim Oral Interview
3. Final Report

These three Epics have multiple Stories assigned - each representing a complex goal then split into small sub tasks achievable within 1 to few weeks.
This was then represented through a Gantt chart that can be seen on the next page:

CE301 GANTT CHART

PROJECT TITLE	Developing A mobile app using machine learning for cat breed recognition	AUTHOR	Olaf Bogus
START DATE	10/10/2022	OTHER	N/A

Task ID	Linked Issue	TASK TITLE	TASK TYPE	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
C301092-2	N/A	Challenge Week Objectives	Epic	Week 1	Week 1	5 days	100%
C301092-3	C301092-2	Create a Risk Assesement	Story	Week 1	Week 1	5 days	100%
C301092-4	C301092-2	Background Reading Summary	Story	Week 1	Week 1	5 days	100%
C301092-5	C301092-2	Technical Deliverables	Story	Week 1	Week 1	5 days	100%
C301092-8	C301092-2	Project Plan	Story	Week 1	Week 1	5 days	100%
C301092-9	C301092-2	Summer Preparation Record	Story	Week 1	Week 1	5 days	100%
C301092-41	C301092-2	Prepare a Presentation	Story	Week 1	Week 1	5 days	100%
C301092-42	C301092-2	GUI Sketches	Sub-Task	Week 1	Week 1	5 days	100%
C301092-43	C301092-2	Background Reading	Sub-Task	Week 1	Week 1	5 days	100%
C301092-6	C301092-2	Algorithm	Sub-Task	Week 1	Week 1	5 days	100%
C301092-7	C301092-2	Create a GUI	Sub-Task	Week 1	Week 1	5 days	100%
C301092-11	C301092-2	Basic Cat Detection	Sub-Task	Week 1	Week 1	5 days	100%
C301092-18	C301092-2	Simple GUI	Sub-Task	Week 1	Week 1	5 days	100%
C301092-22	C301092-2	Functionality Implementation: Image Processing	Sub-Task	Week 1	Week 1	5 days	100%
C301092-10	N/A	Road to Week 11	Epic	Week 2	Week 11	9 Weeks	100%
C301092-11	C301092-10	Machine Learning	Story	Week 2	Week 11	9 Weeks	100%
C301092-12	C301092-10	App Development	Story	Week 2	Week 11	9 Weeks	100%
C301092-14	C301092-11	Advanced Cat Detection	Sub-Task	Week 4	Week 4	1 Week	100%
C301092-19	C301092-12	GUI Improvements Week 4	Sub-Task	Week 4	Week 4	1 Week	100%
C301092-15	C301092-11	Simple Cat Breed Classification	Sub-Task	Week 5	Week 5	1 Week	100%
C301092-23	C301092-12	Functionality Implementation: Breed Search Screen	Sub-Task	Week 6	Week 6	1 Week	100%
C301092-20	C301092-12	GUI Improvements Week 6	Sub-Task	Week 6	Week 6	1 Week	100%
C301092-16	C301092-11	Interim Cat Breed Classification	Sub-Task	Week 7	Week 7	1 Week	100%
C301092-24	C301092-12	Functionality Implementation: Scan History Screen	Sub-Task	Week 8	Week 8	1 Week	100%
C301092-52	C301092-11	TensorFlow to CoreML	Sub-Task	Week 8	Week 8	1 Week	100%
C301092-17	C301092-11	Advanced Cat Breed Classification	Sub-Task	Week 9	Week 9	1 Week	100%
C301092-20	C301092-12	GUI Improvements Week 9	Sub-Task	Week 9	Week 9	1 Week	100%
C301092-25	C301092-12	Functionality Implementation: Breed Search	Sub-Task	Week 9	Week 9	1 Week	0%
C301092-26	C301092-12	Functionality Implementation: Scan History	Sub-Task	Week 9	Week 9	1 Week	0%
C301092-21	C301092-12	GUI Improvements Week 10	Sub-Task	Week 10	Week 10	1 Week	100%
C301092-27	C301092-12	Cat Breed Data Gathering & Database Assembly	Sub-Task	Week 10	Week 10	1 Week	0%
C301092-28	N/A	Road to Final Report	Epic	Week 16	Week 30	14 Weeks	75%
C301092-29	C301092-28	App Work	Story	Week 16	Week 27	9 Weeks	99%
C301092-33	C301092-28	Report Work	Story	Week 22	Week 29	7 Weeks	33%
C301092-34	C301092-28	Open Day Preparation	Story	Week 23	Week 30	7 Weeks	100%
C301092-68	C301092-29	Machine Learning: Switch Datasets	Sub-Task	Week 16	Week 16	1 Week	100%
C301092-70	C301092-29	Machine Learning: Expand Dataset	Sub-Task	Week 17	Week 17	1 Week	100%
C301092-64	C301092-29	Functionality Implementation: Cat Encyclopedia	Sub-Task	Week 18	Week 20	3 Week	100%
C301092-63	C301092-29	Functionality Implementation: History Screen	Sub-Task	Week 19	Week 20	2 Weeks	100%
C301092-85	C301092-33	First Report Plan	Sub-Task	Week 20	Week 20	1 Week	100%
C301092-31	C301092-29	Machine Learning Training	Sub-Task	Week 21	Week 24	4 Week	100%
C301092-30	C301092-29	Visual Improvements	Sub-Task	Week 22	Week 22	1 Week	100%
C301092-32	C301092-29	Bug Fixes	Sub-Task	Week 22	Week 22	1 Week	100%
C301092-92	C301092-29	User Testing	Sub-Task	Week 23	Week 23	1 Week	100%
C301092-93	C301092-29	Cat Detection Feasibility	Sub-Task	Week 23	Week 23	1 Week	100%
C301092-35	C301092-33	Improved Report Plan	Sub-Task	Week 24	Week 24	1 Week	100%
C301092-30	C301092-34	Poster Creation	Sub-Task	Week 24	Week 24	1 Week	100%
C301092-90	C301092-34	Abstract Creation	Sub-Task	Week 24	Week 24	1 Week	100%
C301092-96	C301092-29	User Testing Part 2	Sub-Task	Week 24	Week 24	1 Week	100%
C301092-33	C301092-98	Chapter 1: Project Introduction	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-99	Chapter 2: Project background research	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-100	Chapter 3: App Design Description	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-101	Chapter 4: App Implementation Description	Sub-Task	Week 24	Week 30	6 Weeks	20%
C301092-33	C301092-102	Chapter 5: Machine Learning: Detection	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-103	Chapter 6: Machine Learning: Classification	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-104	Chapter 7: Project Management	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-105	Chapter 8: Conclusion & further work	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-29	C301092-106	Further Improvements	Sub-Task	Week 25	Week 30	6 Weeks	0%
C301092-39	C301092-33	First draft of the report	Sub-Task	Week 25	Week 25	1 Week	0%
C301092-38	C301092-33	Planning out the report	Sub-Task	Week 25	Week 25	1 Week	0%
C301092-65	C301092-29	App Publication	Sub-Task	Week 27	Week 27	1 Week	0%
C301092-40	C301092-33	Final Version of Report	Sub-Task	Week 29	Week 29	1 Week	0%
C301092-37	C301092-33	Create a Planning Record	Sub-Task	Week 30	Week 30	1 Week	0%
C301092-36	C301092-34	Demonstration Planning	Sub-Task	Week 30	Week 30	1 Week	0%

Challenge Week: Week 1

Term 1 - Oral Interview: W 2 W 3 W 4 W 5 W 6 W 7 W 8 W 9 W 10 W 11

Term 2 - Report & Open Day: W 16 W 17 W 18 W 19 W 20 W 21 W 22 W 23 W 24 W 25 W 26 W 27 W 28 W 29 W 30

Figure 1: Gantt-chart(1)

1.3 Report Structure

The main report is divided into 7 chapters. It has 4 elements before the report starts and 2 elements after it ends.

The first chapter covers general project information to introduce the reader into the basic ideas of the project. The second chapter covers background research be it academic papers or articles and videos. Chapters three gives an in-depth review of the app design. Chapters four describes in detail work related to machine learning. Chapter five goes over evaluation of the app and the machine learning models. Chapter 6 focuses on planning and organisation of the project. The last chapter - seven covers the summary and conclusions.

Chapter 2: Project Background Research

2.1 Machine Learning

Extensive research was required to successfully achieve all goals planned for the project. The research was partly focused on purely academic papers and partly on articles and tutorial videos shared on the internet.

With Machine Learning being a very important part of the project I had to do extensive research to learn and understand how to implement and train a model correctly. To prepare for this task I first had to learn and understand mechanics of Machine Learning for Image Classification. To do this I studied papers such as "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review"[1] which gives a very detailed and in depth review of Convolutional Neural Networks used for image classification.
I understood that accuracy would be very important for my project as a low accuracy model is not particularly useful. This topic was very well covered in multiple papers mainly being: "ImageNet Classification with Deep Convolutional Neural Networks "[2], "Going deeper with convolutions"[3], which both focus on improving existing CNN models accuracy. They study what has a direct and positive impact on model accuracy.
Another paper discussing accuracy of a model is "Very Deep Convolutional Networks For Large-Scale Image Recognition"[4] and "Deep Residual Learning for Image Recognition"[5] which also cover neural network depth and the impact it has on accuracy. They both try to find ways to solve the issue of accuracy degradation with increasing depth. Model scaling is a very important aspect of Neural Network Models. Variables that might have impact on accuracy in smaller models will not always stay significant when deployed to large scale models. This was discussed and analysed in "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks"[6].
These academic papers would form the foundation for my Machine Learning Model to classify cat breeds. I would use the gained knowledge throughout the projects duration.

After the autumn term began my main source of new information would become articles and videos explaining various Machine Learning topics rather than academic papers. This was mainly due to time constraints and the complexity of the papers requiring long studies. These papers were reviewed before the end of Challenge Week. To learn about my own specific challenge which was machine learning for cat breed classification I searched for other cat breed classification related projects.
One of my initial aims for the Challenge Week was to create a model that would classify a photo depending if there was a cat present or not. "Cat or Not - An Image Classifier using Python and Keras" [7], "Creating and Deploying a Cat-Dog Image Classifier using TensorFlow and Streamlit" [8], "Building a Cat Detector using Convolutional Neural Networks — TensorFlow for Hackers" [9], "Using TensorFlow to recognise Cats and Dogs"[10], "Cats vs Dogs Classification (with 98.7% Accuracy) using CNN Keras – Deep Learning Project for Beginners" [11] and "How to Classify Photos of Dogs and Cats (with 97% accuracy)" [12] would allow me to see different methods and convolutional layers used for this issue.
Due to time constraints I wasn't able to complete this model during Challenge Week but would later successfully implement it during the Autumn Term. I thought that a simple model only containing two classes would allow me for easier implementation of a more complex model for the breed classification.
During Challenge Week it became obvious after all the research that the model I would use would consist of a Convolutional Neural Network using transfer learning. I would refer to TensorFlow API Documentation to study functions and variable meanings. For the majority of my project I used Google Colab for model training.

Once the Cat Binary Classification model was implemented I could start working on my Breed Classification model. First I searched for a dataset. Finding datasets for Cat Breeds proved difficult as I could only find 2 datasets. The Oxford-IIIT Pet Dataset[13] and Cat Breeds Dataset[14]. The second dataset consisted of 67 different cat breeds and a total of 127 thousand images rather than the first one which had only 12 different breeds and a total of 2371 images.
The more expansive dataset would be used until spring term. After careful study of the 12 Kaggle Dataset[14] page notebooks I tried to create a model of my own as well as "Cat Breed Classification System using CNN - Nurafiqah Mohtar"[15].
I decided to use InceptionV3 as "InceptionV3 with ROC- AUC 92%"[16] used it to achieve the highest accuracy.
My main struggle was converting raw file data into a dataset. It would take me quite some time to solve this issue, I tried following "Build a Deep CNN Image Classifier with ANY Images"[17] and "Dog Breed Classification Using Tensor-flow Keras | Building an Image Classifier"[18] but solved it finally using Pandas DataFrame as implemented in "InceptionV3 with ROC- AUC 92%"[16]. This notebook would prove the most valuable to the early stages of my project.
Once the model was running I struggled with low accuracy and spent a long time researching possible solutions to improve accuracy and fight overfitting.

Seeing as Cat Breed classification was not a very commonly tackled problem I turned to researching Dog Breed classifiers as they worked the same just on a different dataset. "Dog Breed Classification using CNNs"[19] and over 10 notebooks from "Dog Breed Identification" [20] Kaggle page were studied to try and change variables, add different layers and convolutional bases but nothing worked. "Overfit and underfit"[21], "Don't Overfit! — How to prevent Overfitting in your Deep Learning Models"[22] and "Early Stopping to avoid overfitting in neural network- Keras"[23] would also prove fruitless.
This issue would not be solved until the spring term where I studied the dataset I was using and found a lot of inconsistencies in the dataset. This caused me to switch to "The Oxford-IIIT Pet Dataset"[13]. This dataset was much smaller but the data is much more authentic. The overfitting was solved as it was caused by a inconsistent dataset with a large amount of falsely assigned images.

To expand the Oxford dataset I manually searched for photos using Google Images to assemble a dataset of 20 breeds with 200 images per breed.
In order to apply the Keras model in the iOS app I had to familiarise myself with CoreML and exported the models to coreML format. To understand CoreML better I studied different informational sources like "Train a Machine Learning Model with CreateML | Super Simple" [24].

2.2 App Development

My familiarity with iOS devices was the reason I chose iOS as the platform I would develop for. However I had no previous experience developing for iOS, using SwiftUI or even Xcode. This would prove difficult at times but thanks to extensive research I was able to accomplish everything I planned.
A very important part of this was Apple Developer Documentation containing detailed explanation of every function, variable and concepts as well as instructions for using Xcode software.

During Challenge Week I was tasked with creating a prototype of the application. As coding is a complex process video tutorial were key in my ability to create anything.
At the beginning of the Challenge Week my main idea was to create my app as a Progressive Web Application. I researched them thoroughly but I had issues grasping the concept of PWAs and decided to create a native iOS App.
To familiarise myself with Xcode I studied various tutorials and articles.
I studied a lot of different tutorials to grasp how SwiftUI works “Let's create a Simple Login Screen for iPhone in Xcode (SwiftUI)”[25] and “SwiftUI 4: New Photo Picker Tutorial – WWDC 2022”[26]gave me a better idea of how that language functions.
That allowed me to create a functioning app prototype which included a photo picker that took input from camera and photo library as well as on screen messages that updated dynamically after the image upload.
Next I had to learn how to deploy the app to the phone and thanks to “Build and Run app from Xcode onto Actual Device in Xcode 12.15.1”[27] I was able to do just that.
After challenge week I focused on improving the app visually as well as getting it ready to work with an actual Machine Learning model. “SplashScreen for iOS in SwiftUI Tutorial 2022 (Xcode)”[28] would help me understand how to create and implement a launch screen in the app.
In order to implement additional functionality screens in the future I had to research how to implement a bottom navigation bar. This was possible after learning through “How to create a Bottom Navigation Bar with TabView in Xcode (SwiftUI / iOS)”[29] and reading through documentation.
When the bottom navigation bar was finished I had to learn how to actually do anything with the photo uploaded by the user. I had to redesign the PhotoPicker previously implemented. This was accomplished after a careful study of the following sources: “Swift for Beginners: Select Photo from Library iOS (2020)”[30], ”SwiftUI Photo Picker - Compressed Images, UIViewControllerRepresentable”[31], “HOW TO ADD IMAGES TO IOS APP - Swift (2020)”[32], “Tutorial 31: How to Upload and Display Image on iOS App | HOW TO ADD IMAGES TO IOS APP Swift 2021”[33], “iOS 15 Photo Picker Tutorial (2021, Xcode 13, iOS 15) - iOS for Beginners”[34], “Importing an image into SwiftUI using PHPickerViewController – Instafilter SwiftUI Tutorial 9/12”[35], “Wrapping a UIViewController in a SwiftUI view – Instafilter SwiftUI Tutorial 5/12”[36].
I was finally able to pass the uploaded Image as an object to the app and display it back to the user.
With the term coming to the end it was time for me to implement the Machine Learning model in the actual app. This seemed difficult at first but after extensive research involving :” Load CoreML Model into Xcode | Super Simple”[37], “Use CoreML Model in SwiftUI”[38], “Compile CoreML Model in Xcode”[39], “Test CreateML Model with CoreML in an App | Super Simple”[40] and ”CoreML Basics in iOS (Swift 5, Machine Learning, Xcode 12) - 2022 iOS Development”[41] the problem proved easy to solve.

At this point the PhotoPicker still needed some tweaking. It was improved after watching the following: “My Images 1: Photo Picker and Camera in SwiftUI”[42], “SwiftUI Popup Image Picker - Custom Popup's, Image Picker Modifier - Xcode 13 - SwiftUI Tutorials”[43], “The Complete Guide for Integrating Camera and Photo Library in SwiftUI”[44].
To polish off the app visually I have created an app icon thanks to “How to Create an App Icon (2019)”[45] and improved the launch screen after watching “How to Create Launch Screen / Image in Swift 5 and Xcode 11”[46]. This would mark the end of development during the Autumn term.

When worked resumed in the new year, 2 major functionalities were still not implemented. Both of them tackled data management.
To implement the History screen and what came with that - a database and saving the images I studied the Core Data Documentation. The process of learning how to save objects in a Core Data database and creating a database in the first place was tumultuous but made easier thanks to “Saving Images Using Core Data”[47], “Creating a Core Data Model”[48], “Fetching Records With Core Data: Type Methods”[49], "Fetching Objects”[50], “How to combine Core Data and SwiftUI”[51]. Special credit has to be given to “Saving Images Using Core Data”[47] from which some code was used in my project. Mainly the UIImageTransformer extension.
To implement the Encyclopaedia, a very important source for this implementation was “SwiftUI Dynamic List App (Xcode 12, 2021, SwiftUI 2.0) - iOS Development”[52] more minor source “Xcode 11.4.1 - Loading a LaunchScreen Image from Assets Folder”[53]. A lot of the information gained from the History screen was used to implement the encyclopaedia, it just used a different way of storing data.

Conducting the User Tests meant implementing changes desired by the users. One of them being improving the History page management by adding delete entries function and compressing photos before saving.
To accomplish this I studied the following articles: “How to delete Core Data objects from SwiftUI views”[54], “How to delete a Core Data object”[55], “Swift 3 Core Data Delete Object” [56], “Deleting from a Core Data fetch request”[57], “How to delete Core Data SwiftUI?”[58], “Delete/Reset all entries in Core Data?”[59], “SwiftUI update of deleted CoreData entities from different Tab”[60], “How to Save, Fetch, Update and Delete data from Core Data using Swift”[61], “Core data with swiftui (create, read, update and delete)”[62], “Core Data: Quickest way to delete all instances of an entity”[63], “Core data object not nil after deleting from context”[64], “How to delete Core Data objects from SwiftUI views”[65].
In order for the photo files to be stored efficiently I tasked myself with compressing the images prior to saving. This was accomplished with the help of: “How to get the size of data present in coredata store?”[66], “Compress a UIImage”[67] and “In Swift, how to reduce an image file size to a specific size? e.g 1MB”[68].

With User Testing came a lot of visual fixes. These sometimes proved troubling. Thanks to the information I was able to gather on the internet I was able to successfully implement them. The following resources were studied for this purpose: “Date() formatting in SwiftUI”[69], “How to display date and time in SwiftUI”[70], “How to change Background Color of Button in SwiftUI”[71], “SwiftUI Label tutorial – how to create and use Label in SwiftUI”[72], “Introducing SwiftUI TabView and Working with Tab Bar Customization”[73], “Links in SwiftUI”[74], “SwiftUI Button Tutorial: Customization”[75], “How to get file size using FileManager + formatting”[76], “How to create a tappable button“[77], “How to run some code when state changes using onChange()”[78].

Chapter 3: App Implementation

3.1 Interface Design

Summer 2022

The first design was created during summer of 2022. The planning of the app has just began at that point. The original functionality planned for the app included: a classification screen that would work as the home screen - it would use a machine learning algorithm that would classify a photo input from the user and display the results of the classification, a search screen to look through a built in encyclopaedia and a history screen that would allow to access previously classified photos.
These functionalities remained the same throughout the projects duration. All of them were fully and successfully implemented.

The initial design from Summer 2022 was a very rough sketch. At this stage no color pattern was decided on.

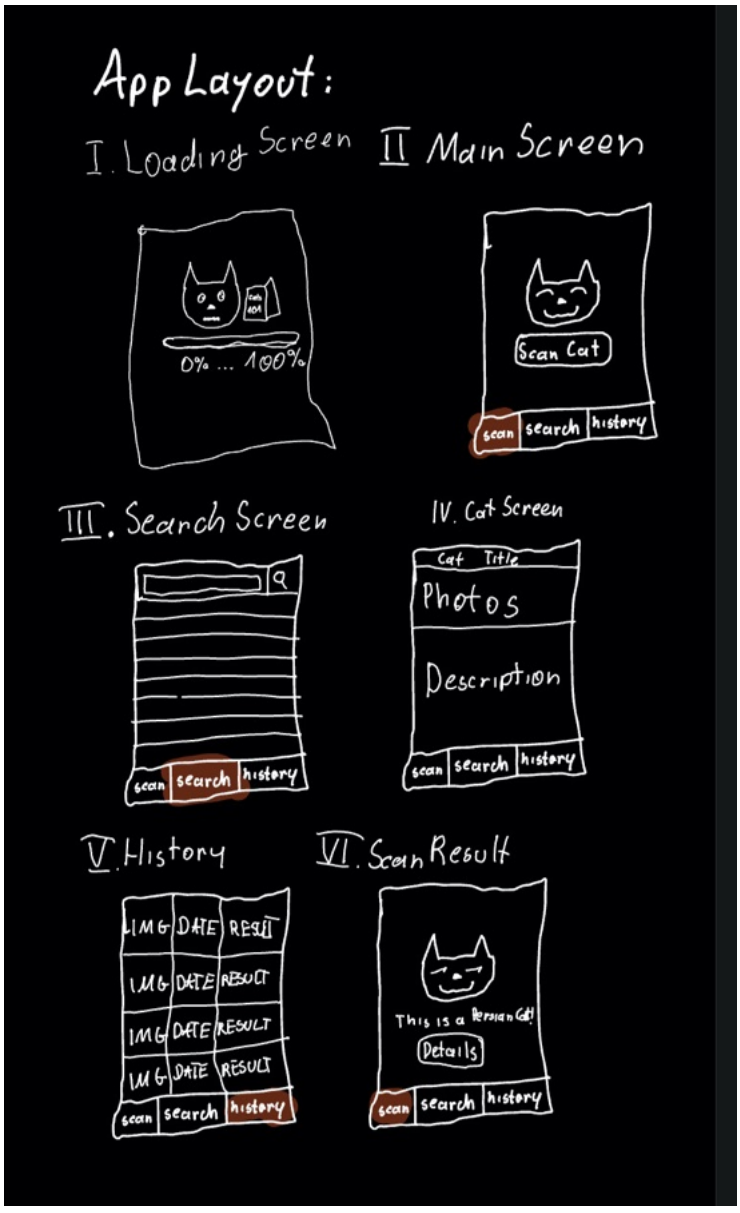


Figure 2: Early stage app design sketches

Challenge Week 2022

The first actual implementation of the app was during the 2022 Challenge Week - when I was tasked with creating a prototype. At this stage the app was just supposed to be able to take an input and display a result. The result would be based on a randomly generated boolean. The prototype was purposefully made to look very raw.

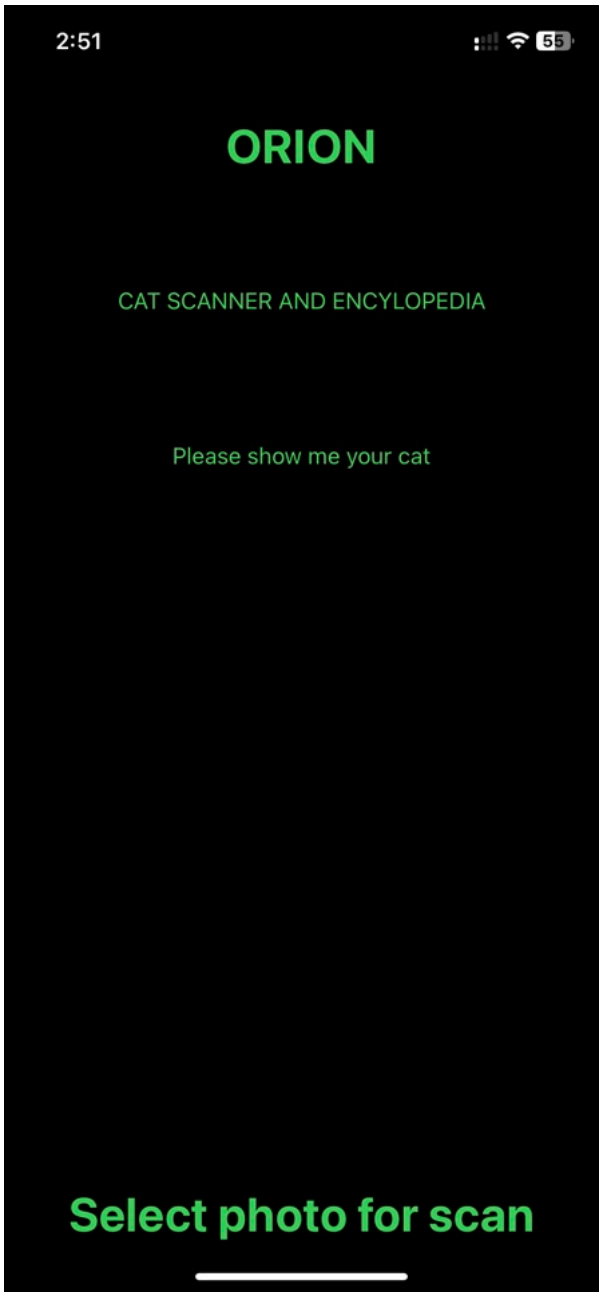


Figure 3: Prototype Home Screen (No input)

Figure 4: Prototype Home Screen (Output)

The future versions of the app would be based on this design, greatly improved at each step.

Autumn Term 2022

During the autumn term the app design would forego the most essential changes yet. The color scheme was decided to be centred around coral orange. Coral would be the background color throughout the app and all the elements displayed on top of it would we soft cream pink or yellow. The app would see it's first launch screen with an icon holder sign for where the future icon would be placed. The app now actually used a machine learning model for classification and displayed the top 3 results and the matching percentage for the top 1 result. During the autumn term the two remaining functionalities (encyclopaedia and history tab) were not fully implemented yet. That would stay the case until spring term.

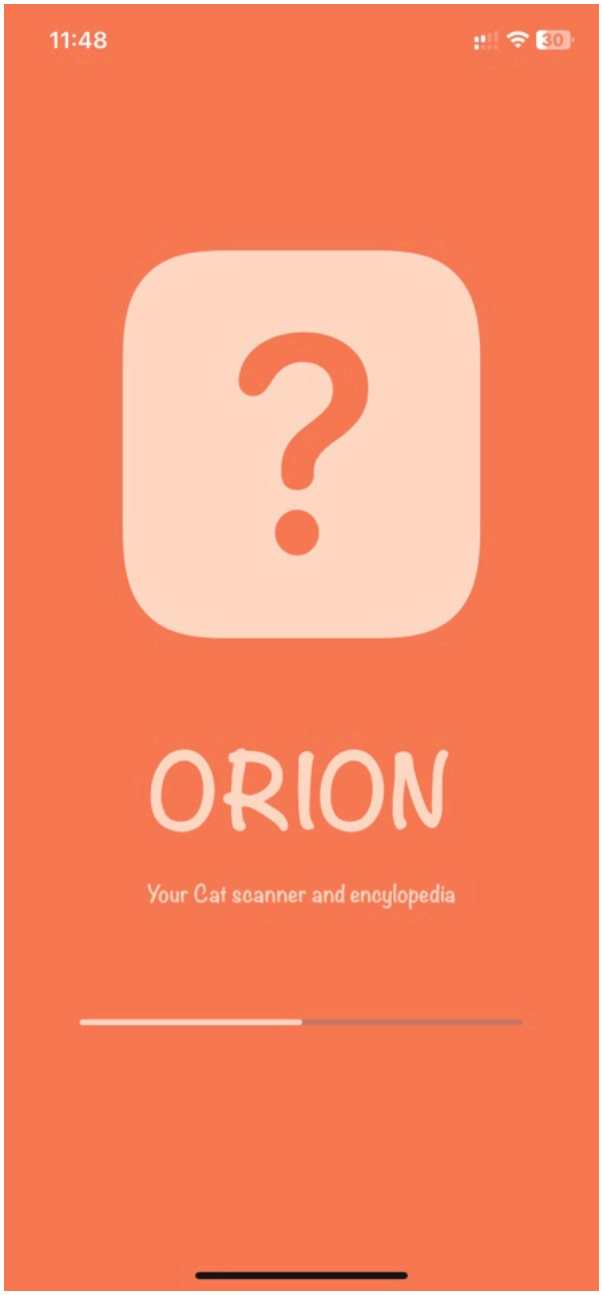


Figure 5: Autumn Term App Version Launch Screen

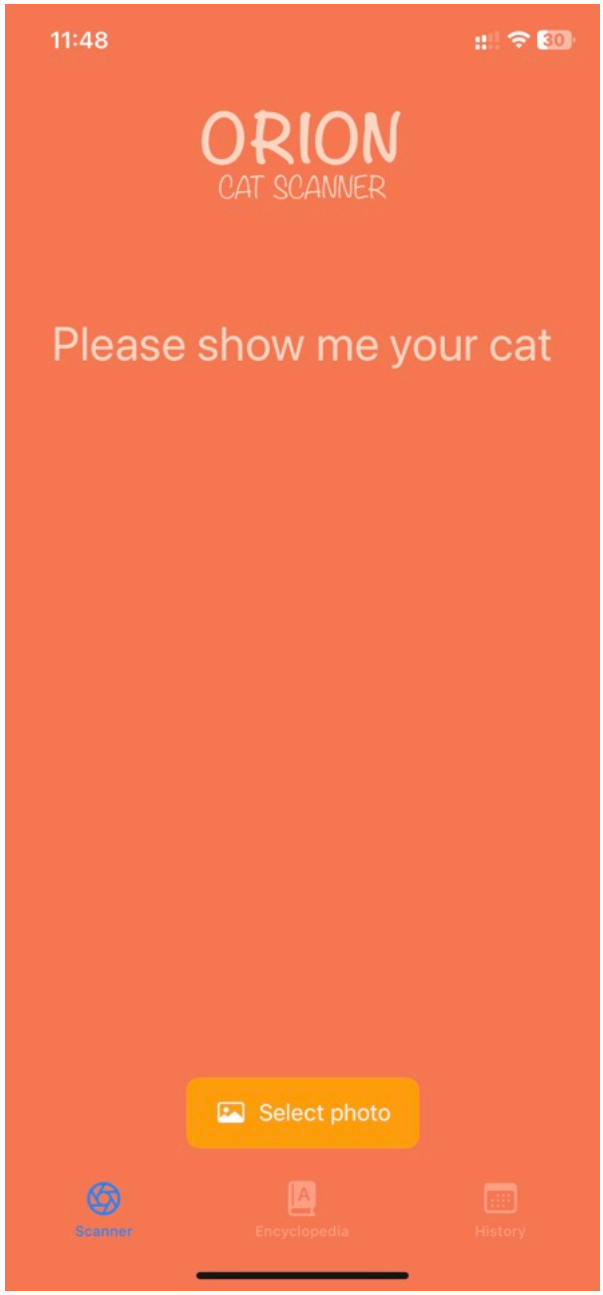


Figure 6: Autumn Term App Version Home Screen

Figure 7: Autumn Term App Version Result Screen

Spring Term 2023

The Spring term saw the completion of final polishes to the app. Taking it from an unpolished interface with a lot of potential into a professional grade and ready to release product.
All the missing functionality would be implemented with a fully functional encyclopaedia, history screen and the classifier page utilising 2 machine learning models.
The whole app interface would see essential changes that improved the visual appeal as well as accessibility. The app became much more intuitive to use without the need for complex manuals. The aim was for everything to be self explanatory.
User Testing was essential in this process. The different background of the users(both android and iOS users as well as people of different ages and those suffering visual impairments) ensured that the app was friendly for everyone. Their different points of view came up with key changes that were not obvious for me as the creator.
The details of user testing are covered in the evaluation chapter.

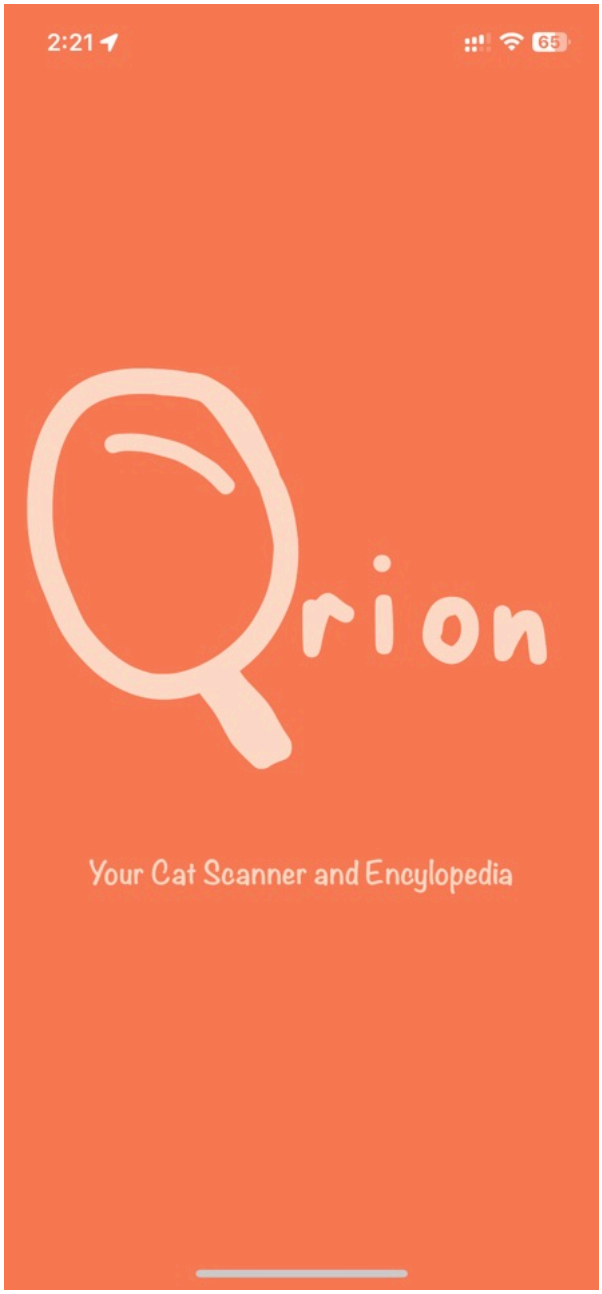


Figure 8: Final App Version Launch Screen

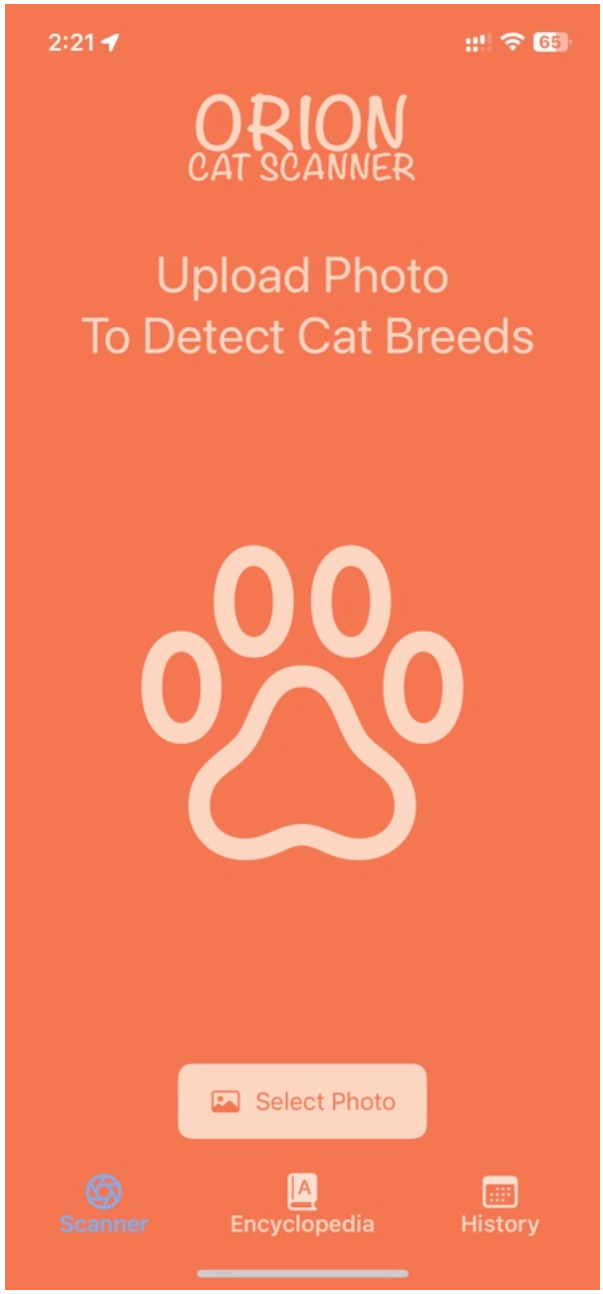


Figure 9: Final App Version Home Screen

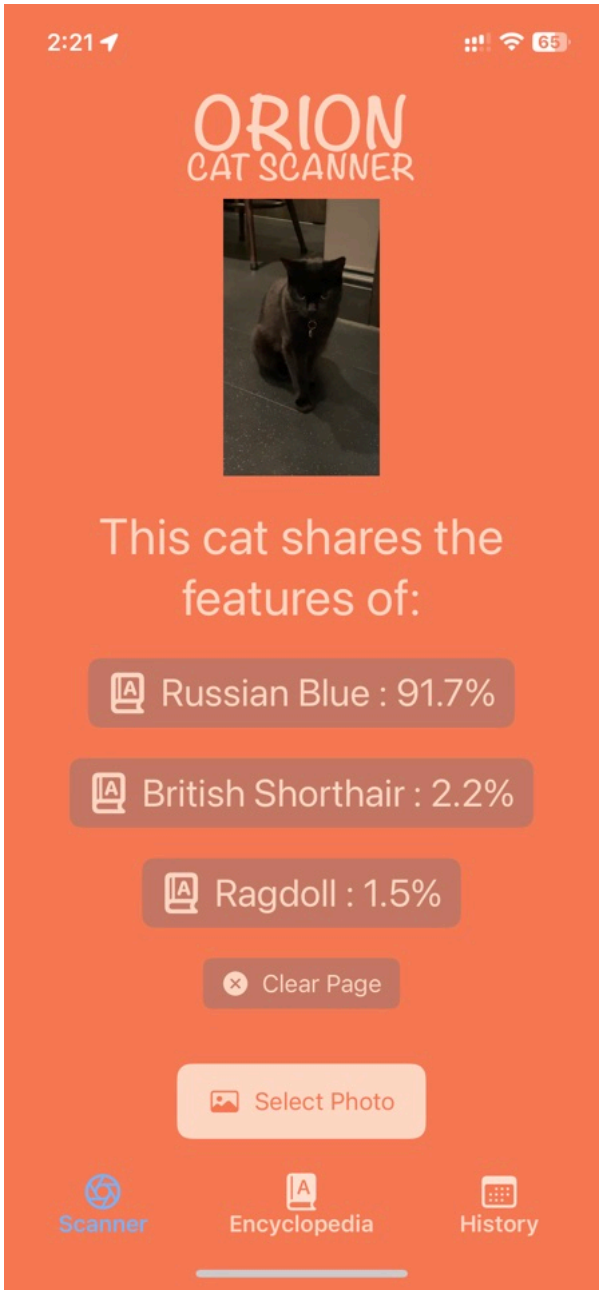


Figure 10: Final App Version Result Screen

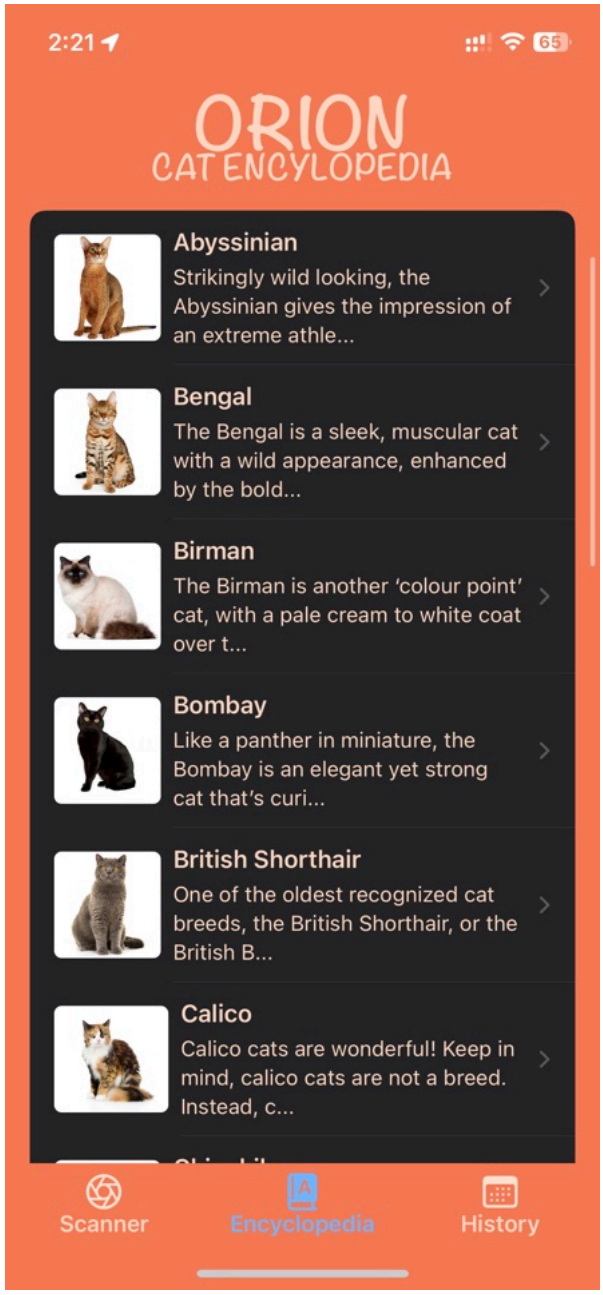


Figure 11: Final App Version Encyclopaedia

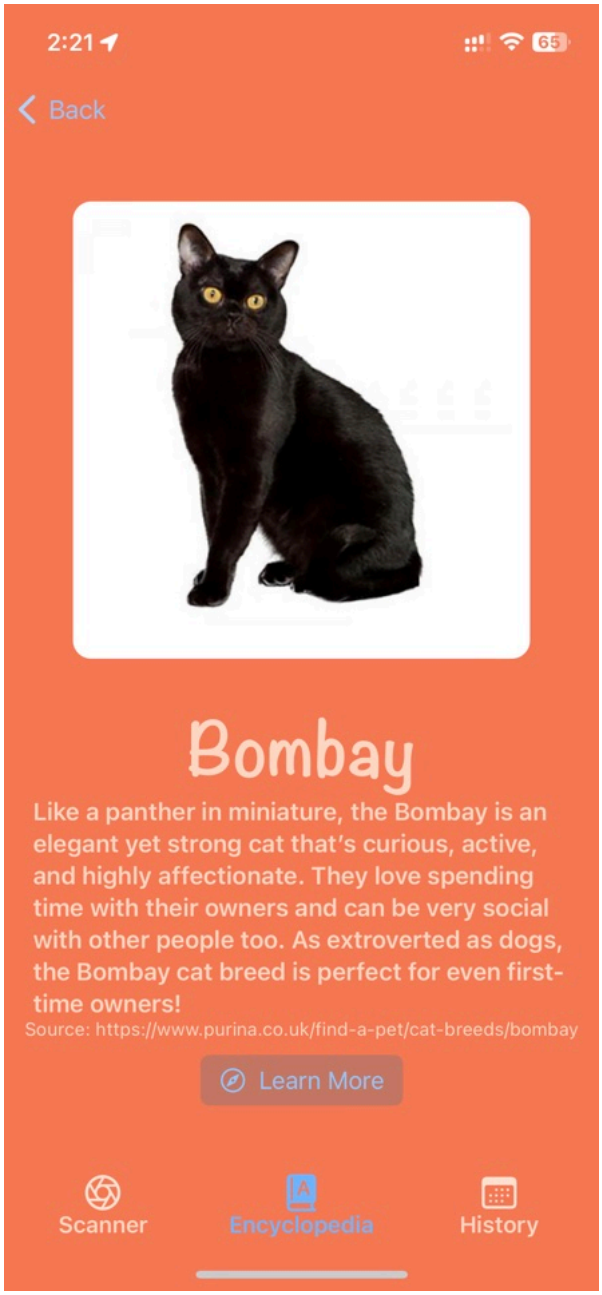


Figure 12: Final App Version Encyclopaedia Entry Page

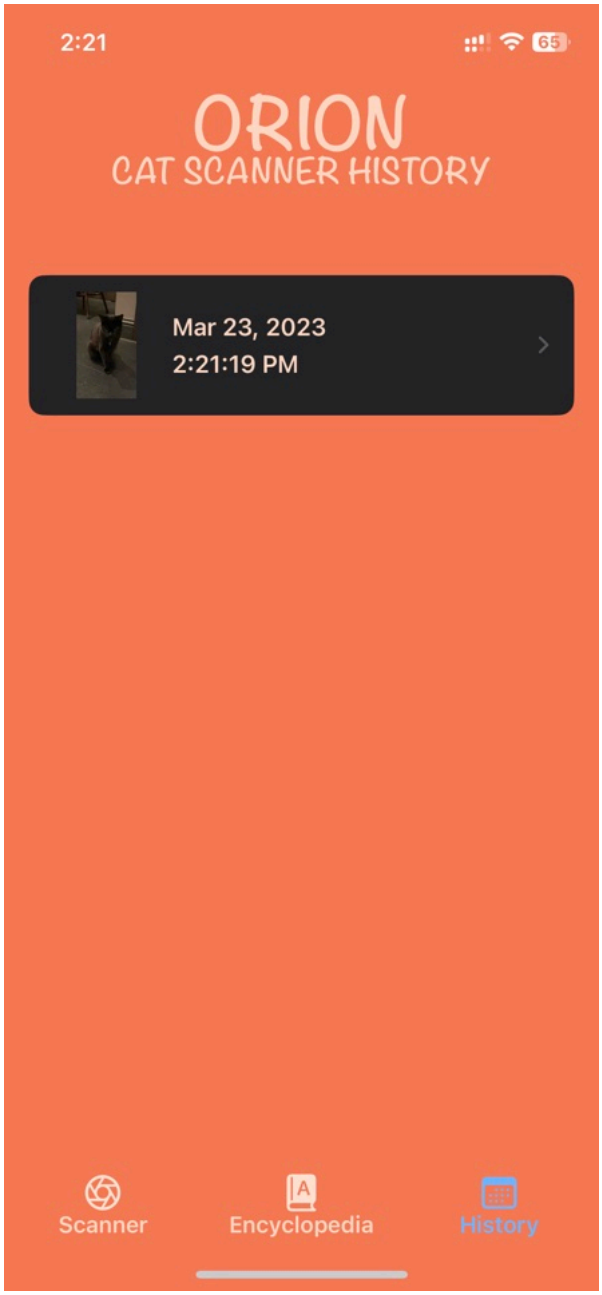


Figure 13: Final App Version History Page

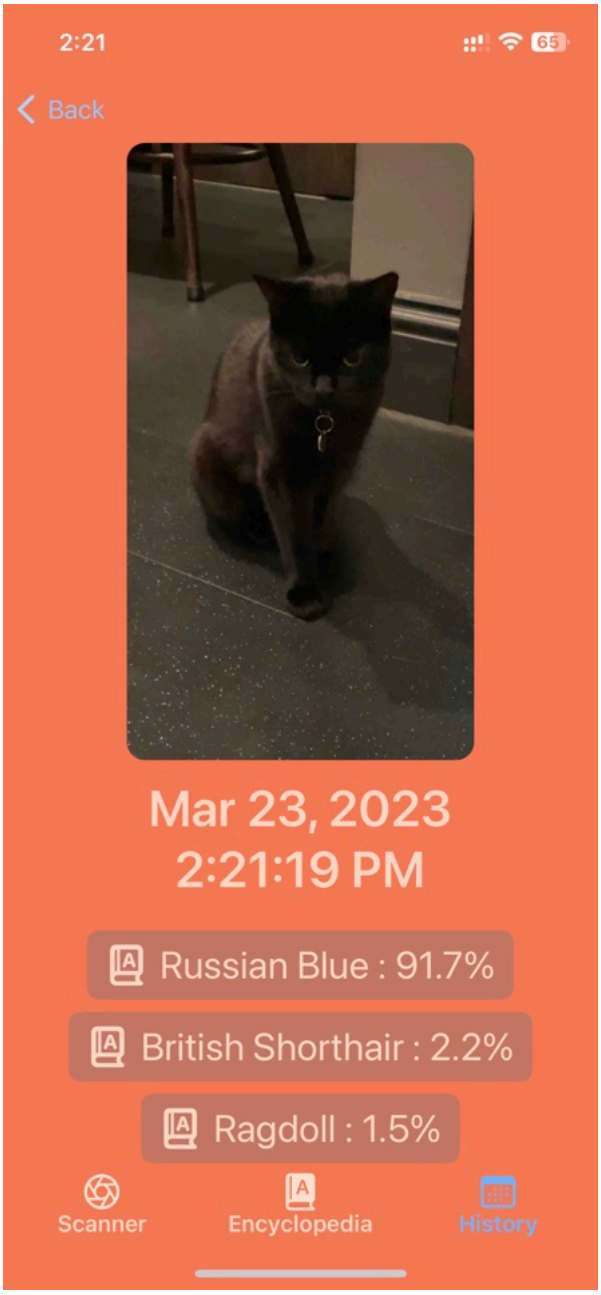


Figure 14: Final App Version History Entry Page

Design Summary

During the duration of the project the interface design has seen a tremendous amount of growth. Every implemented change was made with the user experience in mind.
The aim was to design an app with visual appeal, a friendly user interface and accessibility. The necessary time was taken to ensure every button and every sign is noticeable to the user and self explained in its purpose.
The initial sketches from summer 2022 laid the foundation for the interface. On that foundation the whole app was created and designed. With immense improvements and development. Thanks to the thorough user testing, functionality testing, case testing and intensive research I was able to make informed decisions developing the app of which results speak by itself.

3.2 Code Design

The app was developed using Apple's Xcode 14.0 - 14.3 with SwiftUI. It has been designed for iOS and optimized for iPhones with 6.1 inch screens.

The program classes and structs are split into 4 categories with a single sub category:

- Main
- Screens
    - Item Views
- Extensions
- Data Managers

Brief Summary

The main category is simply the main struct from which the whole app is run. It only loads the ContentView which contains the TabView with the three main bottom tabs.

Screens are any structs that present a different screen to the user. These include the Classifier, Encyclopaedia, History as well as the sub screens presenting single Encyclopaedia or History item. Referred to as Item Views.
Every screen includes a header which consists of two lines. The first line simply states Orion. The second line corresponds to the tab the user is in at the moment: Scanner, Encyclopaedia or History.

Figure 15: App Header (Scanner)

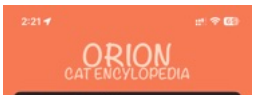


Figure 16: App Header (Encyclopaedia)

Figure 17: App Header (History)

Screens have itemViews - these are detailed screens for items presented in the tables in the main screens. itemViews do not have a header.

Extensions are classes or extensions that support functionality of the app implemented in other structs.

Data Managers are used for the purpose of storing data.

Category Overview

MAIN

As previously mentioned the main struct simply invokes the ContentView.

SCREENS

Content View

The ContentView is the main struct that is always running and invokes other structs through the TabView that is presented by the ContentView. Due to the simplicity of the code I am not presenting any snippets. It is available in the repository.

Classifier

Classifier acts as the home page and is used for processing and classifying photo inputs from the user. This struct is the most complex part of the whole App as it invokes multiple functions and objects.

The three functions are:

getBreed

This function takes a String and returns a BreedData object corresponding to the String. The String is the Breed name, it then iterates through all BreedData to find a breed with that name. It Is then returned.

detectCat

This function uses the classification model to decide if there is a cat on the input photo. Depending on the result it sets the boolean state variable to true or false.

classifyCat()

This changes the display variables (messageOut1-4).
It first checks the boolean returned by the detectCat function and sets the display message accordingly.
If there is a cat detected on the photo the message would be “This cat shares the features of:” for the opposite result the message is “I don’t think this is a cat but if ti was it would be:”.
Then it passes the photo through the breed classification model, after the output is returned it is sorted and passed into strings in the following format: Breed name + “:” + match confidentiality value and percentage sign.
After setting the strings the function compresses the photo and saves the photo and classification outputs as well as the time stamp to the database. In the end the display labels are set to the previously formatted strings.

The View of the Classifier consists of the previously mentioned header that every main screen has.
The Classifier View can be in two states that is determined by the currently selected photo that can either be null or not null.
If the photo is null - the View is in Home Page mode. The Home Page mode displays a welcome message, an icon and a button prompting user to select an input.
If the photo is not null - the View is in output mode. The output mode displays the user input photo and three buttons each corresponding to the top 3 results going from 1 at the top to 3 on the bottom. Each button has a Encyclopaedia icon followed by the classified breed and the confidence percentage. Each button is clickable as it is a NavigationLink that leads to the Encyclopaedia page corresponding to the breed.
This is implemented with a single conditional statement.

```
if image == nil {
    Spacer()
        .frame(height: 10)
    Text("Upload Photo \n To Detect Cat Breeds")
        .foregroundColor(Color.init(red: 255/255, green: 217/255, blue: 197/255))
        .font(.largeTitle)
        .multilineTextAlignment(.center)
        .scaledToFit()
    Spacer()
    Image(systemName: "pawprint")
        .font(.system(size: 200))
        .foregroundColor(Color.init(red: 255/255, green: 217/255, blue: 197/255))
        .scaledToFit()
    Spacer()
} else {
```

Figure 18: Code Snippet(Classifier Screen)

Encyclopedia

The Encyclopaedia struct includes the corresponding header. The struct has a single variable which loads the Breed information to be presented in the table. This is then passed into a simple for each table. Photos are fetched from assets by breed name.

```
HStack {
    List(breeds, id: \.id) { breed in
        NavigationLink(destination: BreedView(breed: breed), label:{
            HStack{
                Image(breed.name)
                    .resizable()
                    .scaledToFit()
                    .frame(height: 70)
                    .cornerRadius(5)
                VStack(alignment: .leading, spacing: 5){
                    Text(breed.name)
                        .foregroundColor(Color.init(red: 255/255, green: 217/255, blue: 197/255))
                        .fontWeight(.semibold)
                        .lineLimit(2)
                        .minimumScaleFactor(0.5)
                    Text((String(breed.description[..<String.Index(utf16Offset: 80, in:
                        breed.description)]))+"...")
                        .foregroundColor(Color.init(red: 255/255, green: 217/255, blue: 197/255))
                        .font(.subheadline)
                        .foregroundColor(.secondary)
                }
```

Figure 19: Code Snippet(Encyclopaedia)

Each Item in the table is a NavigationLink that leads to the Encyclopaedia itemView with the corresponding breed. All the Images in the encyclopaedia are sourced from the Purina Cat Library Website[83].

ItemView: EncyclopediaView
The EncyclopediaView is a very simple struct. It is passed a breed when opened. It then fetches the information from that breed and displays the photo followed by the name, description, source and external URL with more information(which is the same as the source). The URLs are lead to the corresponding Purina Cat Library website[82] and the Smithsonian Zoo Website[103].
History

This struct uses just two variables. One being a list storing items fetched from the database, the other being a context to connect to the database.
The struct utilises three functions.

getItems
: This function fetches items from the database

removeEntry
: This function removes an item from the database

getBreed
: This is the same function as in the Classifier struct. It matches a Breed based on the name string.

The View of the History struct includes the corresponding header followed by (just as in the Encyclopaedia view) a table. It also consists of NavigationLinks that in this screen lead to the subview HistoryView which fetches detailed data about the chosen History item.
Each NavigationLink presents a small icon of the saved photo and data stamp of the classification.
The table allows to swipe to the left to invoke the onDelete functionality which then calls the previously mentioned removeEntry function to delete items from the database.
Every time the view is loaded the data is fetched from the database again and displayed in the table. This ensures the displayed data is always up to date.

itemView: HistoryView
The HistoryView is passed a Cat object. It then displays all the information from the object in the following order: photo, timestamp, classification results as clickable Encyclopaedia buttons in order from #1 result to #3 result.

EXTENSIONS

Extension

This extension was created by Afraz Siddiqui from iOS Academy. I studied it in the video “CoreML Basics in iOS (Swift 5, Machine Learning, Xcode 12) - 2022 iOS Development”[60].
It allows for images to be transformed into a buffer and then resized. This is essential as the coreData models require the input to be a resized CVPixelBuffer

Image Picker

This extension was sourced from Jkirst from StackOverflow post “How does the Imagepicker get analyzed by the MLMODEL?,”[79]. This extension facilities the functionality of the ImagePicker by processing the input data from the user and making it accessible by the app.

UIImageTransformer

This extension facilitates image compression in the Classifier struct. It was sourced from “Saving Images Using Core Data”[67].

DATA MANAGERS

BreedData

Breed data holds a list of 20 breeds that are displayed in the Encyclopaedia page. It is fetched by the Encyclopaedia and by iteration loaded in the table. Each breed consists of the breed name, description and url.

HistoryDataManager

This is an automatically generated file upon setting up coreData. It was edited to transform raw data that the photo is saved us back to a photo. It facilitates the initialisation of the database.

Cat

This is an automatically generated file upon setting up coreData. It was edited to add functionality for fetching date information. It can fetch the calendar date with the getDate() function and the time information with the getTime() function.

3.3 Deployment Guide

To Deploy the App on a device the following items are needed:

- a Mac device with Xcode 14 or higher installed
- a iPad or iPhone running iPadOS 16 or higher or iOS 16 or higher
- an active Apple ID
- a suitable cable to connect the device to the computer
- internet access

Step 1
On your device head to Settings then Privacy & Security and scroll to the bottom. Press Developer mode and turn it on. A device restart will be required.

Step 2
Get the latest App code from the gitlab link : https://cseegit.essex.ac.uk/22-23-ce301/22-23_CE301_bogus_olaf_w

Step 3
Connect the device to the computer and Trust the computer on your device.

Step 4
Open the downloaded project in Xcode

Step 5
To your left press the top item.

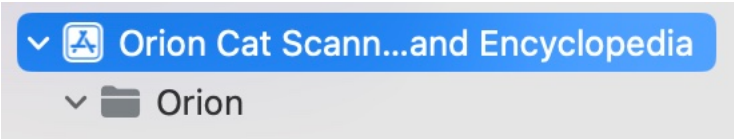


Figure 20: Xcode Instructions (Step 5.1)

Then In the menu select Signing & Capabilities

Make sure the Automatically manage signing checkbox is ticked.
In the field Team you will have to select your existing team or press add an account.
If you are adding an account you will be required to log in with your Apple ID.
After adding an account make sure it is chosen as the Team. Set the name of the bundle identifier.

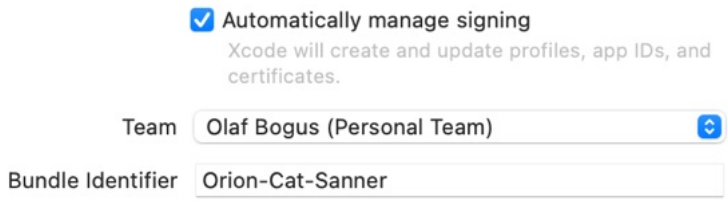


Figure 21: Xcode Instructions(Step 5.3)

Step 6
Unlock your device

Step 7
At the top of the Xcode window it should say “Any iOS Device” press on that and from the list select your device.

Figure. 22: Xcode Instructions(Step 7)

Step 8
Click the run button to your left or go to Product -> Run

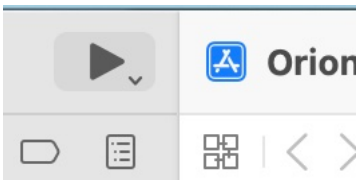


Figure 23: Xcode Instructions(Step 8)

Step 9
When the App is fully installed on your phone you will get a message saying the App author is unknown. You will have to head to Settings -> General -> VPN & Device Management.
From there you have to press on your Apple ID and press Trust.

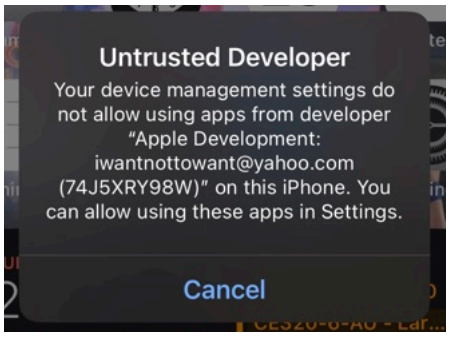


Figure 24: Xcode Instructions(Step 9.1)

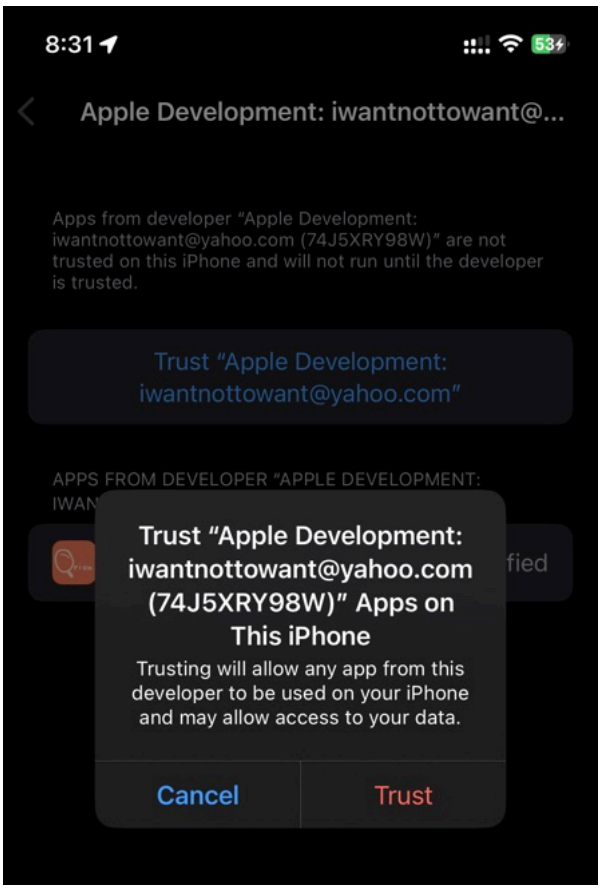


Figure 25: Xcode Instructions(Step 9.2)

Step 10
The app should now be automatically launched. You can disconnect the cable. This will exit the App. Now you can run the App on your device whenever wanted. The app will stay active on the phone for 7 days. After 7 days you will have to repeat the process to sign the app again.

3.4 User Guide

Once the App is installed on the device please press on the icon to launch it.
You can do that from your home screen or App Library.

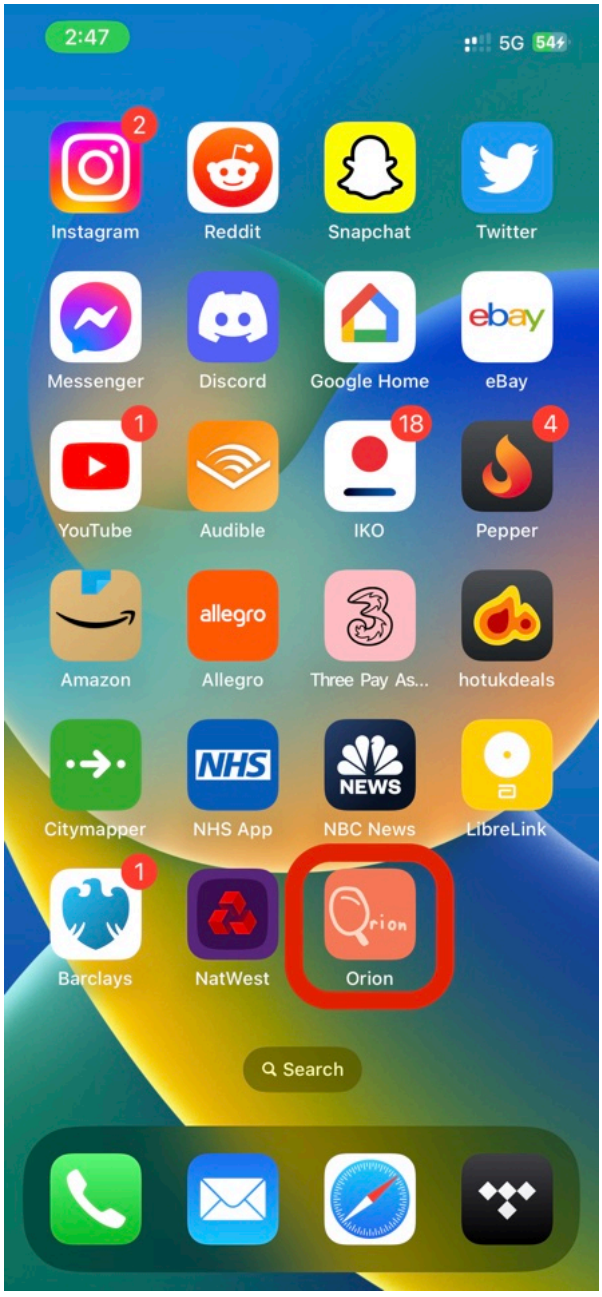


Figure 26: User Guide Launching(1)

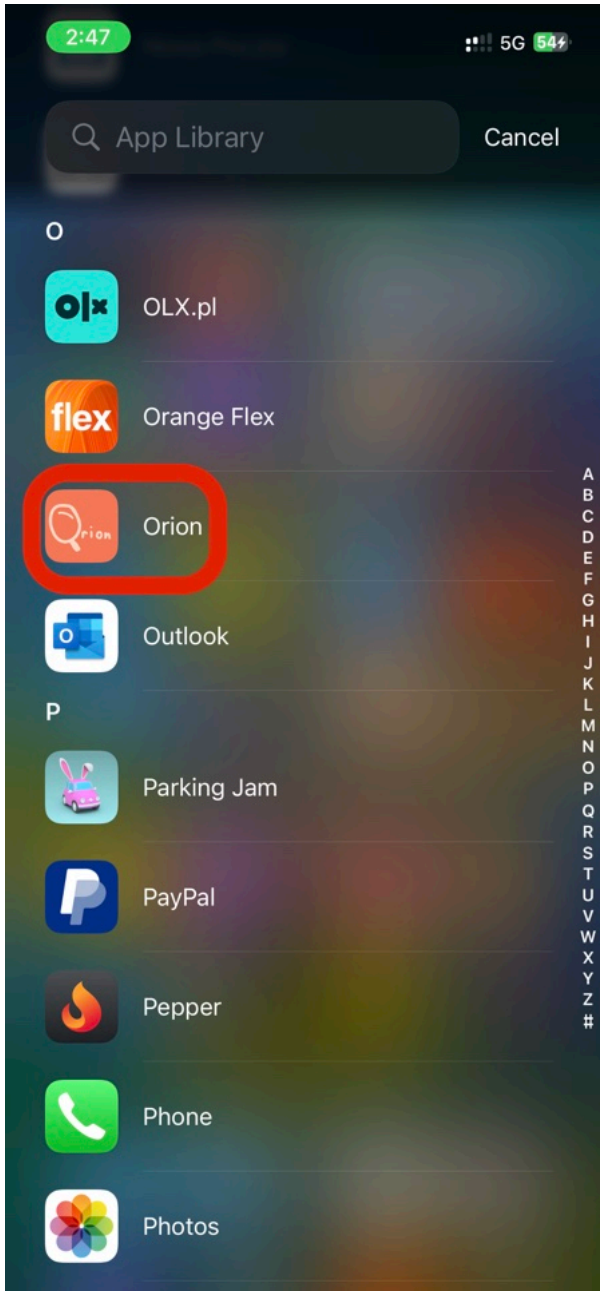


Figure 27: User Guide Launching(2)

Once pressed the App will start launching. You will see a loading screen. Once the App is launched you will see the home screen. On the bottom of the screen you have three tabs. Each one directs you to a different section of the app. You can switch through them freely.

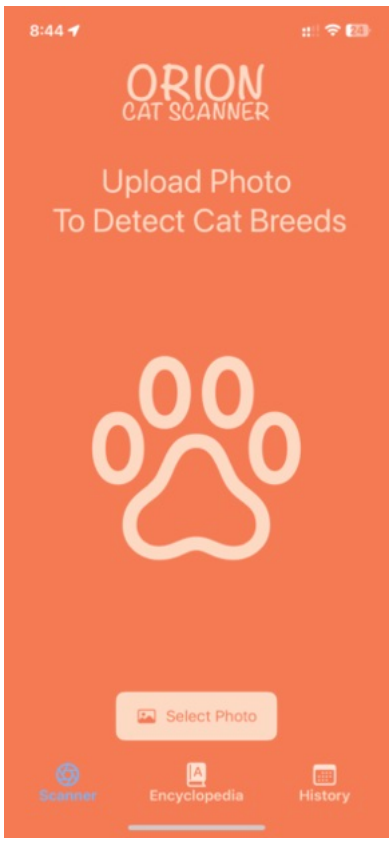


Figure 28: User Guide Home Screen

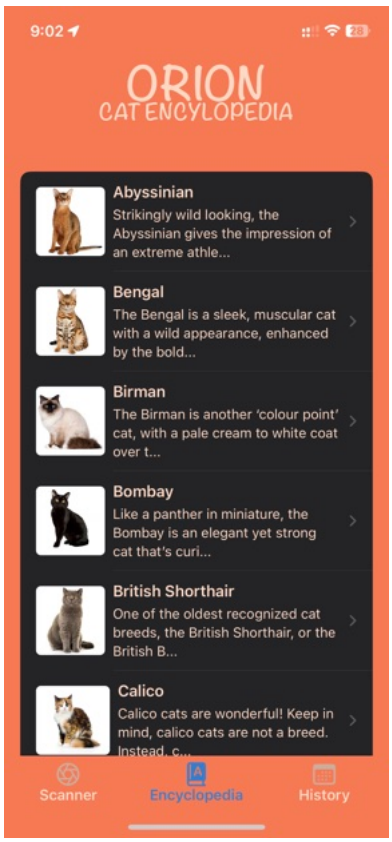


Figure 29: User Guide Encyclopaedia Screen

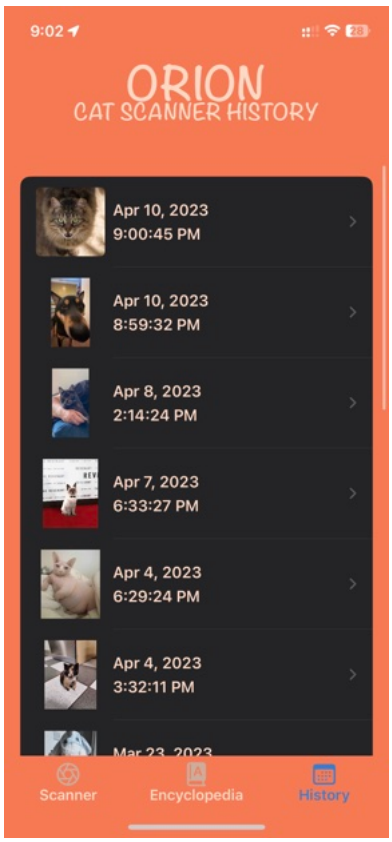


Figure 30: User Guide History Screen

Classifier Tab
In the Classifier you can select a photo for classification.
Upon pressing Select Photo you will get a choice menu. You can either open the Photos Library to pick a photo or the Camera app to take a photo. When selecting from the Library just press on the photo to submit it for analysis. When taking a photo with the camera app press Use Photo after taking it.

Figure 31: User Guide Photo Picker(1)

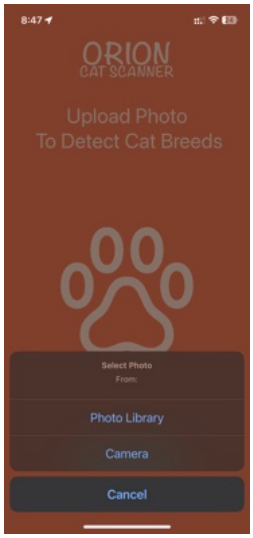


Figure 32: User Guide Photo Picker (2)

Figure 33: User Guide Photo Picker(3)

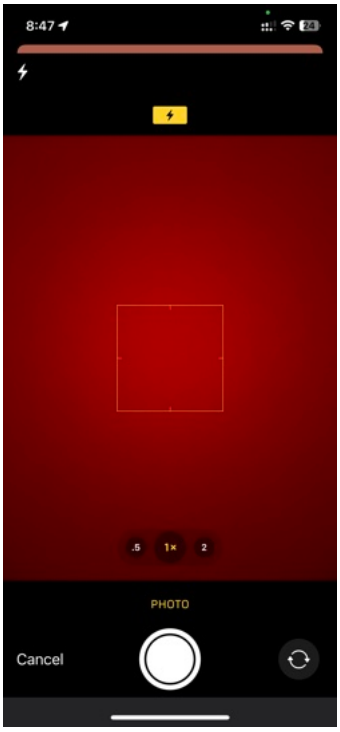


Figure 34: User Guide Photo Picker(4)

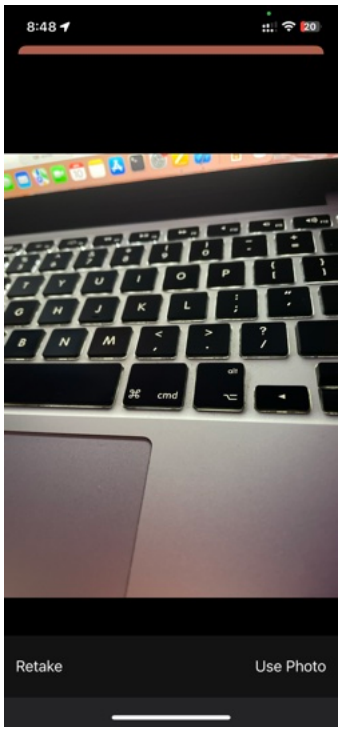


Figure 35: User Guide Photo Picker(5)

Once the input is passed you'll see a result in the following format:

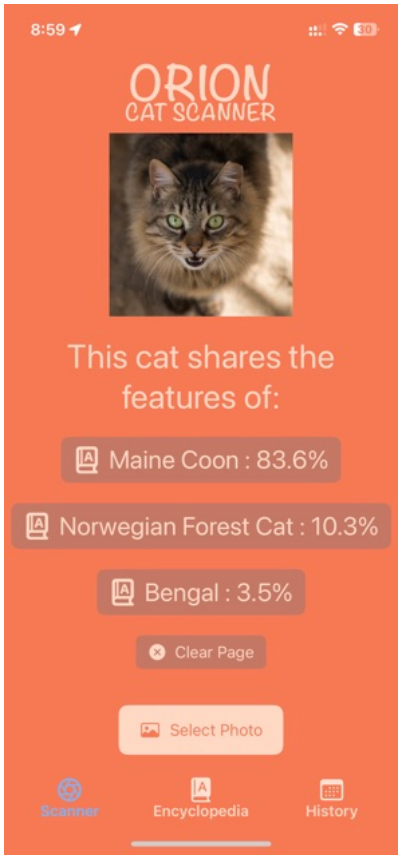


Figure 36: User Guide Classification Results(1)

You are presented with 3 buttons displaying the top 3 results of the classification, a button to clear the classification page to the home page and a button to select a new photo.

When pressed one of the three encyclopaedia icon buttons you will be directed to the Encyclopaedia page with details about the corresponding breed. If you press the “Clear Page” button you will be taken back to the Home Scanner Page.

Figure 37: User Guide Classification Results(2)

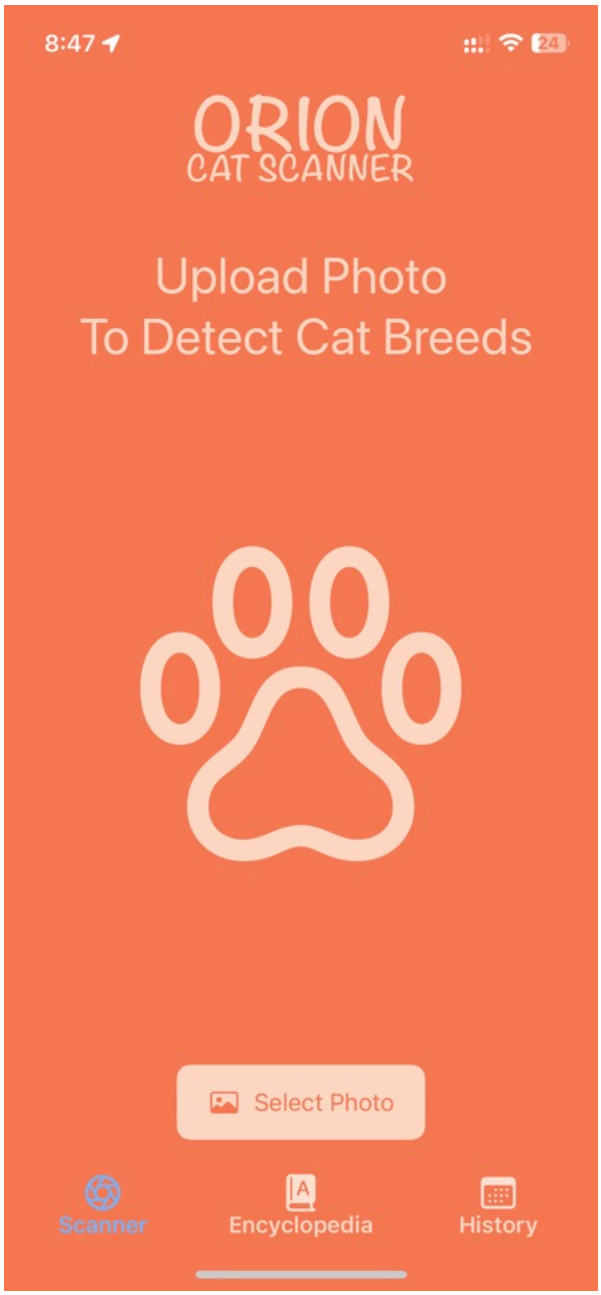


Figure 38: User Guide Classification Results(3)

Encyclopedia Tab

When pressing the Encyclopaedia tab from the bottom navigation bar you will be taken to the Encyclopaedia page.

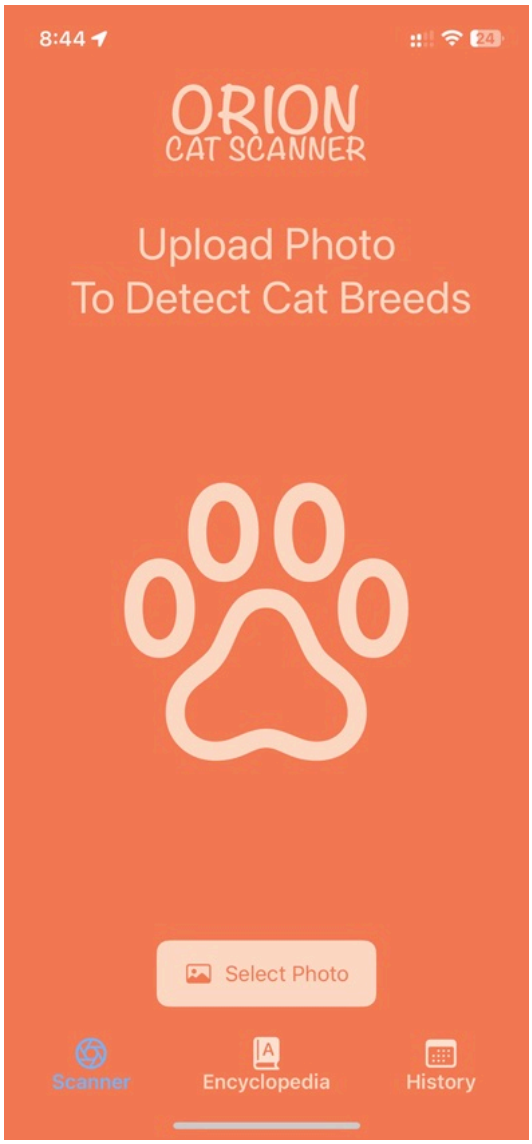


Figure 39: User Guide Encyclopaedia(1)

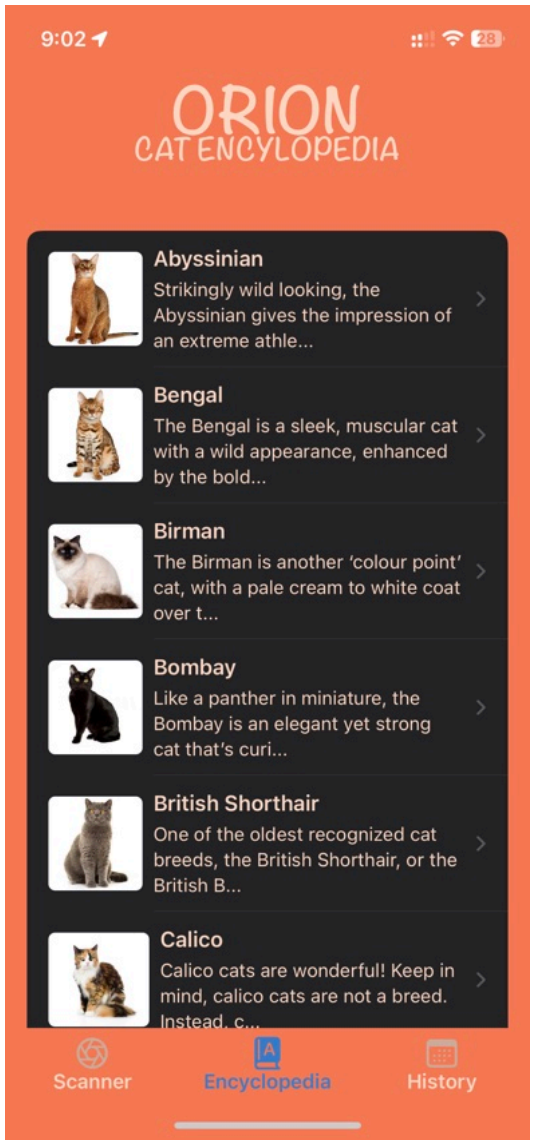


Figure 40: User Guide Encyclopaedia(2)

You can scroll through the Encyclopaedia table. Each item in the table you can press to open the detailed view of the breed.

Figure 41: User Guide Encyclopaedia(3)

The bottom of the detailed view has a Link button that will take you to a website that has expansive information about the breed.

History Tab
The History Tab is very similar to the Encyclopaedia page. Instead of breeds it presents a table presenting your previously scanned photos. You can see a small preview of the saved photo and the date and time it was classified on. You can also swipe on an item to the left to delete it.

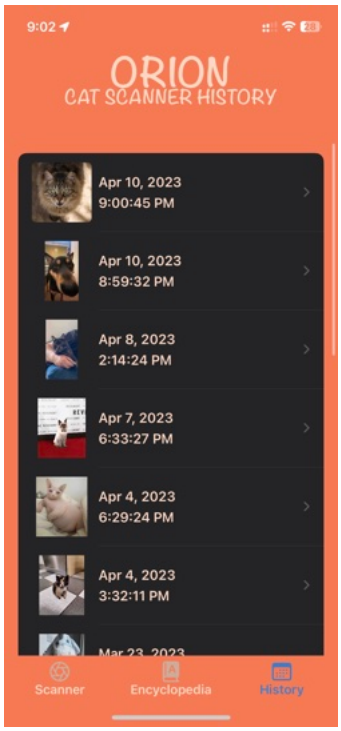


Figure 42: User Guide History(1)

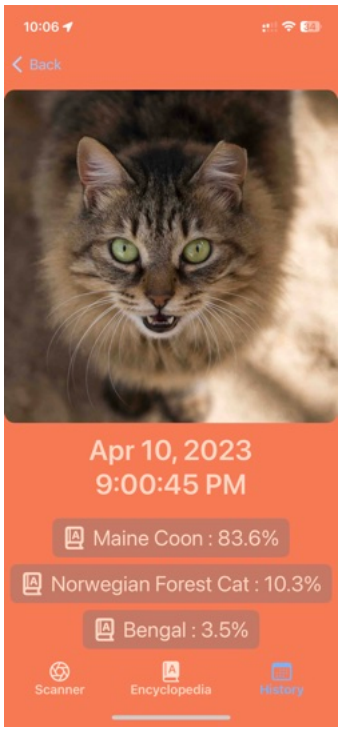


Figure 43: User Guide History(2)

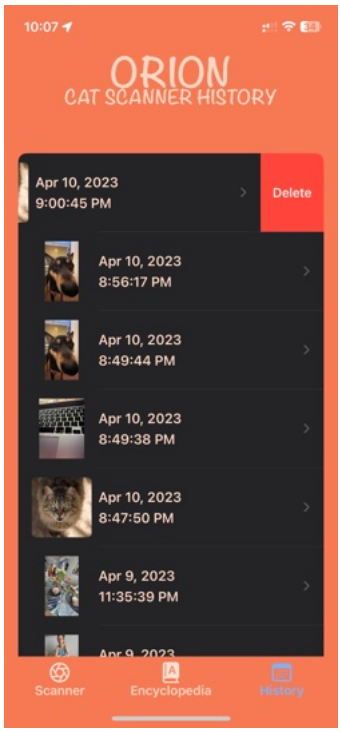


Figure 44: User Guide History(3)

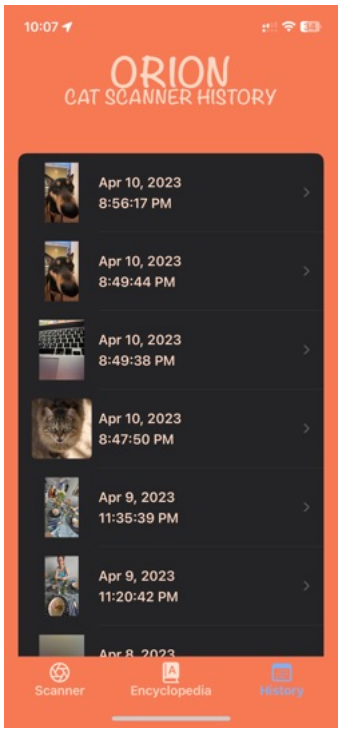


Figure 45: User Guide History(4)

Now you can use Orion: Cat Scanner & Encyclopaedia on your own! Enjoy!

3.5 Application Development Process Summary

Challenge Week

I first started making plans for the app in summer 2022. I made some rough sketches for the interface. With the challenge week I created a prototype with a purposefully rough look. I have never used Xcode or SwiftUI before, trying to apply Java principles in SwiftUI has not always proven easy. During the creation of the prototype I had to re learn a lot of the principles I have been taught. Some tasks were much easier to complete than in Java but some seemed needlessly and overly complicated. Creating the prototype itself was not a very hard task and Xcode made it very simple and straightforward. Creating a button to take inputs from users photo library was the most complicated task that took a considerable amount of time during challenge week. It took effort to get used to the way SwiftUI works.

Autumn Term

The main challenge for development during autumn term was time constraint. All the tasks were accomplishable, the reason I failed to finish everything as planned during autumn term was due to the time.
Machine Learning part of the project took a considerable amount of time that would otherwise be spent on app development. Familiarising myself with Xcode and SwiftUI was straightforward but some tasks would require more time than I anticipated. Thus the encyclopaedia and history implementations had to be delayed to the spring term.
Reworking the prototype took some effort and a lot of thinking. Choosing colours so that they are visually pleasing as well as easily distinguishable and friendly to the user took a lot of trial and error. Polishing the look of the application in ways that were small turned out to be complicated to implement.

Spring Term

The goals of the spring term was to completely finish the app. That meant implementing the functionalities missed previous term and user testing the app to adjust the interface and functionality. The implementation of History and Encyclopaedia screen proved simple enough to program once I had the time.
After careful user testing which consisted of two rounds the application saw major changes. The details of the user testing are described in the evaluation chapter. The user feedback felt very refreshing and eye opening. As someone who designed and programmed the whole application everything seemed very obvious to me. To the users the functionality was not so obvious and needed more clearly visible buttons. Thanks to the testing I was able to fix easy mistakes that were a result of overworking and overlooking like typos and wrong sorting order.

Chapter 4: Machine Learning for Cat Breed Classification

4.1 Cat Breed Classification

4.1.1 Model Design

The model used for classification is a TensorFlow model using Keras. It is primarily based on InceptionV3[29].
The InceptionV3 architecture is visualised below:

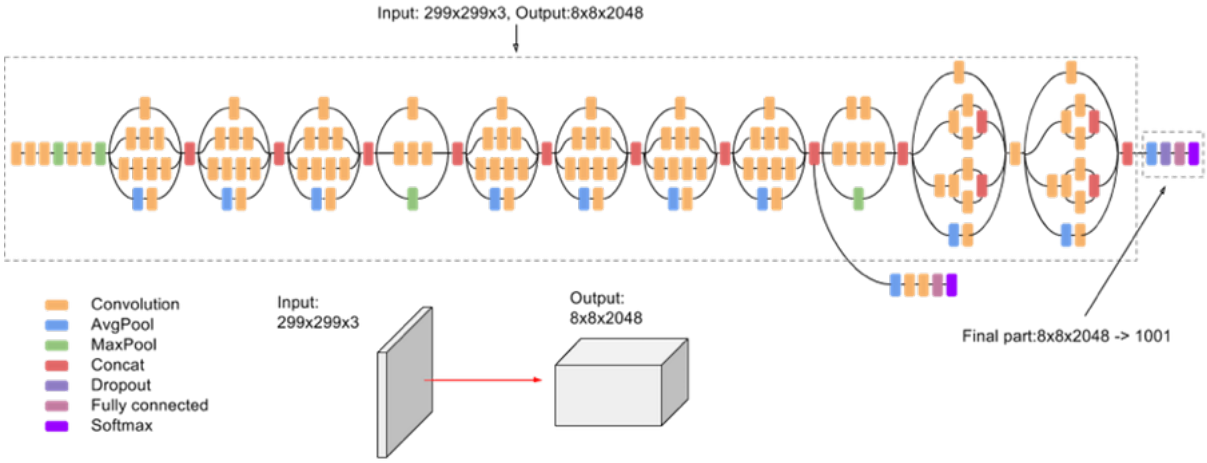


Figure 46: High-level diagram of the InceptionV3 Model[79]
Source: https://cloud.google.com/tpu/docs/inception-v3-advanced

The InceptionV3 is has its layers frozen for the purpose of my model.
For the purposes of increasing accuracy I have added 11 additional layers.

Layer (type)	Output Shape	Param #
inception_v3 (Functional)	(None, 8, 8, 2048)	21802784
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dense (Dense)	(None, 256)	524544
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 256)	65792
dropout_1 (Dropout)	(None, 256)	0
flatten (Flatten)	(None, 256)	0
dense_2 (Dense)	(None, 256)	65792
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 256)	65792
dropout_3 (Dropout)	(None, 256)	0
dense_4 (Dense)	(None, 20)	5140

Total params: 22,529,844
Trainable params: 727,060
Non-trainable params: 21,802,784

Figure 47: Machine Learning Model Summary

The final parameter count was 22529844 of which 727860 were trainable giving a trainable ratio of 30.95%.
This allowed the model to break 90% validation accuracy as well as accuracy in testing.

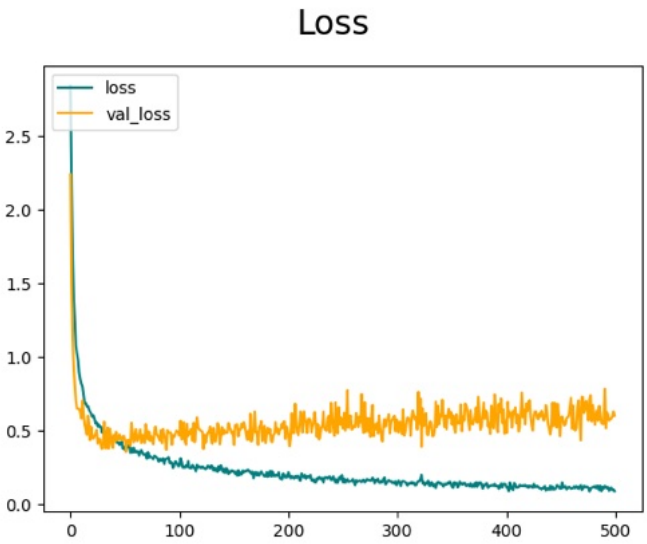


Figure 48: Classification Model Loss

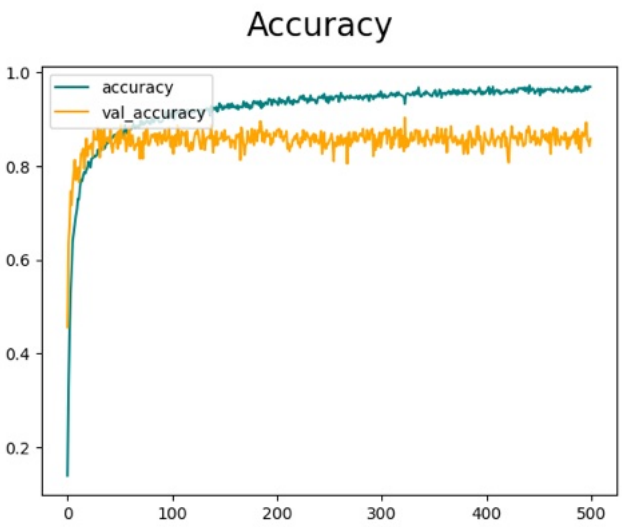


Figure 49: Classification Model Accuracy

4.1.2 Model Development

The first model design consisted of InceptionV3 with frozen layers and 4 additional layers.

```
Layer (type)                 Output Shape
=================================================
inception_v3 (Functional)    (None, 8, 8, 2048)

global_average_pooling2d_3   (None, 2048)
(GlobalAveragePooling2D)

dense_6 (Dense)              (None, 128)

dropout_3 (Dropout)          (None, 128)

dense_7 (Dense)              (None, 10)
```

Figure 50: Original Model Summary

This yielded a total amount of parameters 22,066,346 of which trainable parameters were 263,562 giving a ratio of 1.19%.

The decision to commit to InceptionV3 was made after a review of 3 other convolutional neural networks. Inception V3 seemed to be marginally performing better.

Model Comparison

	VGG16	ResNet50	Xception	InceptionV3
Training accuracy (40 epochs)	24.05%	99.47%	99.80%	99.81%
Validation accuracy (40 epochs)	25.79%	42.37%	49.21%	47.11%
Test accuracy	25.52%	46.06%	51.26%	46.69%
F1 Score	0.10	0.47	0.53	0.46
ROC-AUC	0.50	0.85	0.87	0.83

Figure 51: Model Comparison(1)

As this model was being trained on a faulty dataset different combinations of additional layers were tried. The model topped at peak validation accuracy of 47.11%. This would not improve until the dataset switch. This model design would be the same model design when switching to the correct dataset.

Post dataset switch the same model achieved a validation accuracy of 89.47%.
The goal of breaking 90% accuracy was reached after testing multiple combinations of different additional layers. At last it was reached using the current design

```
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Functional)    (None, 8, 8, 2048)        21802784

global_average_pooling2d (G  (None, 2048)              0
lobalAveragePooling2D)

dense (Dense)                (None, 256)               524544

dropout (Dropout)            (None, 256)               0

dense_1 (Dense)              (None, 256)               65792

dropout_1 (Dropout)          (None, 256)               0

flatten (Flatten)            (None, 256)               0

dense_2 (Dense)              (None, 256)               65792

dropout_2 (Dropout)          (None, 256)               0

dense_3 (Dense)              (None, 256)               65792

dropout_3 (Dropout)          (None, 256)               0

dense_4 (Dense)              (None, 20)                5140

=================================================================
Total params: 22,529,844
Trainable params: 727,060
Non-trainable params: 21,802,784
```

Figure 52: Current Model Summary

This allowed the validation accuracy to peak at 95.52%.
The decision to stick to InceptionV3 was retested at this point. The results seemed to marginally favour that neural network again.

Model Comparison

	ResNet50	Xception	InceptionV3
Training accuracy (40 epochs)	94.80%	91.30%	92.60%
Validation accuracy (40 epochs)	84.27%	86.67%	87.73%
Test accuracy	84.81%	86.08%	86.40%
F1 Score	82.75%	86.51%	85.09%
ROC-AUC	99.47%	99.38%	99.48%

Figure 103: Model Comparison(2)

4.1.3 Model Dataset

The data is passed to the model through an ImageDataGenerator with a batch size of 16. This allows for image augmentation. Each image is resized to a size of 299 pixels by 299 pixels. While supplied to the model the images can be flipped and shifted.

The first dataset used for this model was Cat Breeds Dataset[16]. It consisted of 67 different cat breeds and a total of 127 thousand images. The dataset was quite imbalanced and was never used in its entirety due to the size. First I chose a small subset of the dataset consisting of 12 breeds. Those breeds were: Abyssinian, Balinese, Cornish Rex,
Domestic Medium Hair, Himalayan, Manx, Oriental Short, Russian Blue ,Singapura ,Tonkinese and a total of 452 images.
This was switched to a different subset while struggling with overfitting. The new subset counted 12 breeds as well and 4053 images. The breeds were as follows: Abyssinian, American Bobtail, American Curl, Applehead Siamese, Balinese, Birman, British Shorthair, Burmese ,Cornish Rex ,Egyptian Mau ,Exotic Shorthair and Havana.

During preparation for Interim Oral Interview I spotted a lot of inconsistencies in the images of the dataset. What got my attention was a lot of rabbit photos. After a careful review of the dataset I have concluded that the overfitting was caused by those inconsistencies.

That prompted me to switch to the Cats and Dogs Breeds Classification Oxford Dataset[X]. This dataset consists of 37 labels including dog and cat breeds. Out of the whole dataset 12 labels are cat breeds. Each having 200 images. These being: Abyssinian, Bengal, Birman, Bombay,British Shorthair, Egyptian Mau , Maine Coon, Persian, Ragdoll, Russian Blue, Siamese, Sphynx. Implementing this dataset in the model resolved the issue of overfitting. This dataset brought in a different issue, the size of the dataset is relatively small. I have strived to increase it to 20 labels.

I achieved that by manually sourcing images. These were partly from the previous Cat Breeds Dataset[15] after a considerate visual study of the cat breed and manual review of the photos. After extracting all the appropriate images from the dataset I then turned to the internet to make sure that each breed had 200 images. This was a lengthy and arduous process.

In the end I was able to achieve my goal and expand the dataset to 20 labels with 200 images per label. The final labels were: Abyssinian, Bengal, Birman
Bombay,British Shorthair, Calico, Chinchila, Egyptian Mau , Korat, Lykoi Cat, Maine Coon, Norwegian Forest Cat, Persian, Ragdoll, Rex, Russian Blue, Sand Cat, Scottish Fold, Siamese, Sphynx. All with equal distribution.

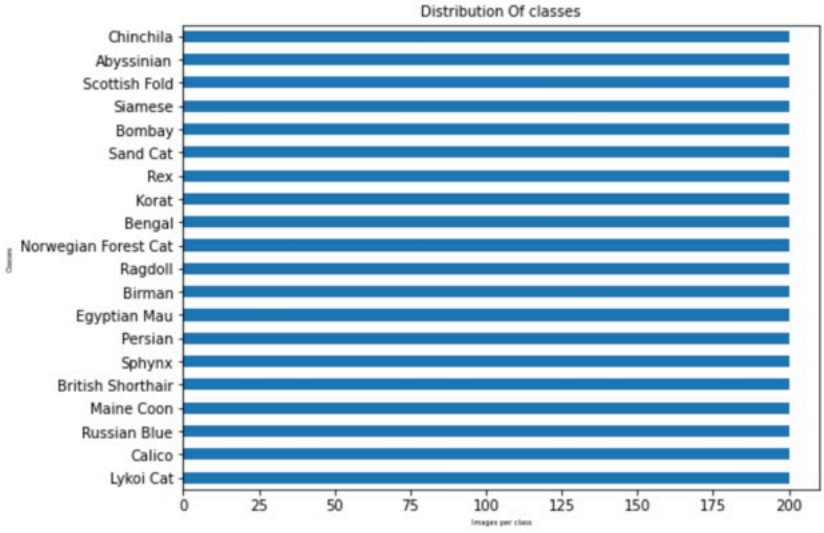


Figure 54: Expanded Cat Breed Dataset Distribution

4.1.4 Keras to CoreML Conversion

In order for the model to be used in the iOS app it had to be converted to a CoreML model. This was a straightforward process. To convert the model an image input type had to be specified and the labels had to be supplied. The whole process was accomplished in few lines of code

```
import coremltools as ct

#Specyfing input shape
image_input = ct.ImageType(shape=(1, 299, 299, 3,),
                           bias=[-1,-1,-1], scale=1/127)
#Supplying labels
classifier_config = ct.ClassifierConfig(labels)
#Conversion
model = ct.convert(
    model,
    convert_to="mlprogram",
    inputs=[image_input],
    classifier_config=classifier_config,
)
```

Figure 55: CoreML export code

4.1.5 Process Summary

During the beginning of the process I have struggled to understand how to correctly implement a machine learning model. Creating datasets from raw data files were very difficult for me to understand. After a long series of failed attempts I was able to implement a model.

The next major issue was over fitting. As described in the dataset analysis the first dataset I chose was not accurate. Later after a more thorough research into the problem I learned that the dataset was created with a tool that downloads all images available for a keyword internet search. That explained why rabbits appeared so often in the image files themselves as a lot of rabbit breed names are similar to cat breed names. Solving this issue was both euphoric and irritating as it was a very easily avoidable mistake that was only made due to my unfamiliarity with the process.

Switching to the Oxford Dataset I aimed to expand it. With expansion the accuracy of the model went down. This prompted me to search for breeds that were visually distinct from other breeds to cause less accuracy degradation. After a laborious process of finding images for the expanded dataset the whole dataset was assembled.

The accuracy was not at the desired level as I wanted the model to have at least 90% accuracy. Due to my limited hardware resources I was training the model on Google Colab. This had it own different limits. The time outs did not allow me for long training.

I then switched to training on my own machine. Even though the training process was much slower on my machine it allowed for much longer continuous training. It turned out that the model was able to break 90% after a more lengthy training process.

4.2 Cat Detection through Binary Classification

4.2.1 Model Summary

Binary classification was implemented with the same model architecture as that used for the breed classification. The dataset was switched to reflect cats and non cat images.

The dataset used for this model was based on Dog and Cat Detection[80] Kaggle dataset. It was then expanded with images of random objects to increase the model accuracy beyond just dog and cat differences. The dataset consists of 8394 images in total of which cat images were 5020 and 3374 non cat images. Due to the simplicity of the model it only needed to be trained for 5 epochs.

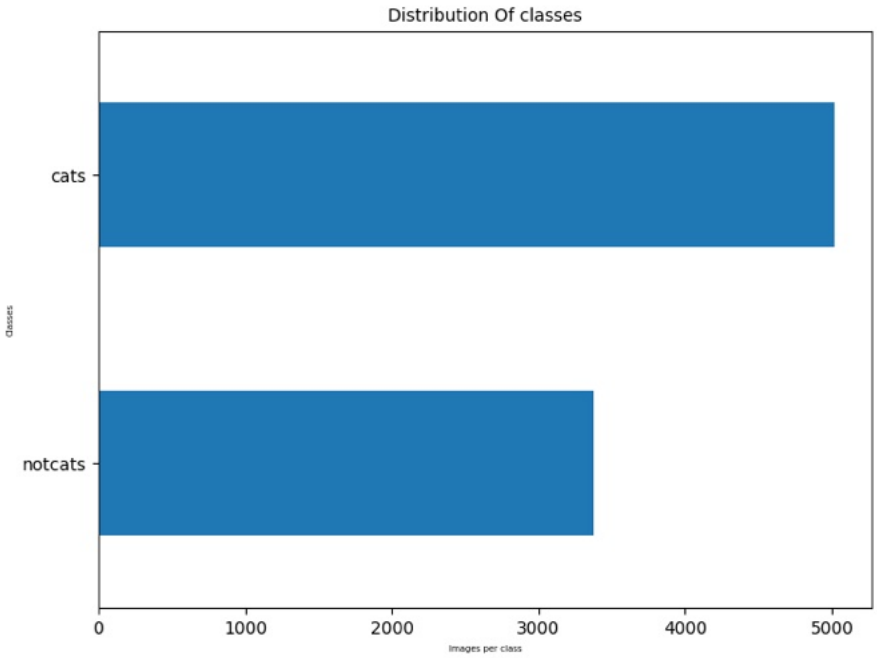


Figure 56: Binary Classification Dataset Distribution

The model was able to achieve over 99% validation accuracy and less than 0.021 loss.

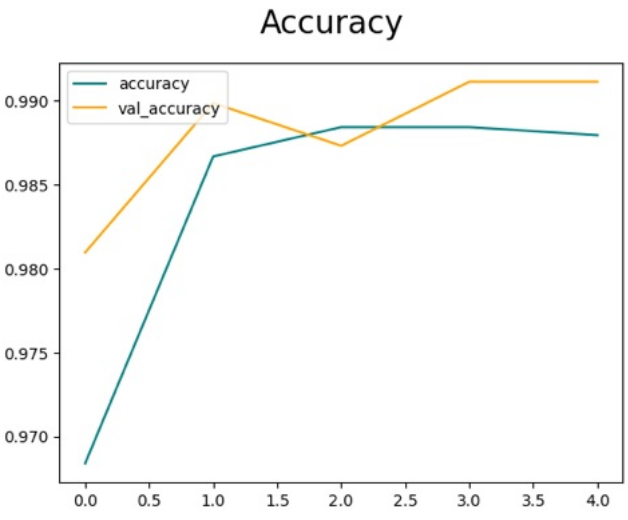


Figure 57: Binary Classification Training Accuracy

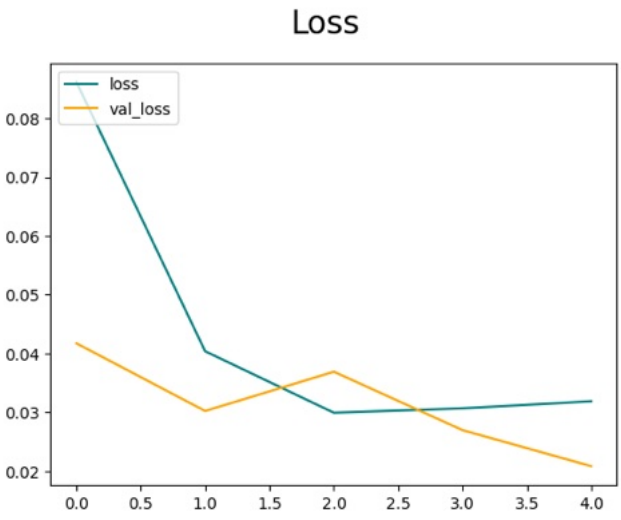


Figure 58: Binary Classification Training Loss

Chapter 5: Evaluation

Chapter 5.1 App Evaluation
Chapter 5.1.1 Case Testing

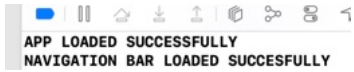
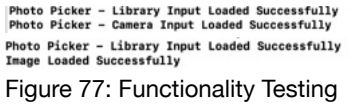
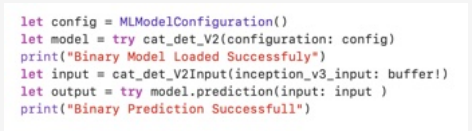
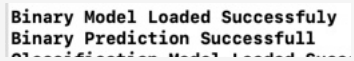
The below table presents use cases that were used to test the app. Every step was replicated manually on the deployed app.

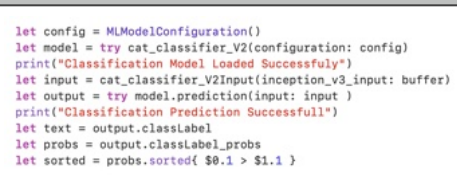
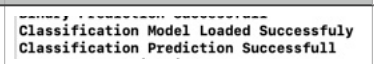
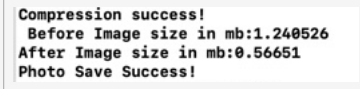
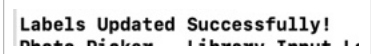
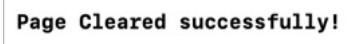
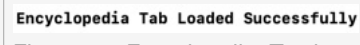
User Story	Task Steps	Desired Outcome	Step Outcome
A veterinarian is seeing a new patient today. In order to create a new health record in the clinic database they have to input the cat breed. The veterinarian is not sure what breed the patient is and the owner has no clue either. The veterinarian needs help determining what breed the new patient is.	1. Vet opens the Orion App 2. Vet presses Select Photo 3. Vet presses Camera in the input menu 4. Vet takes photo of the cat and submits it 5. Vet looks at the classification results and makes their determination	1. The app launches 2. The Photo Picker menu opens 3. The camera app launches 4. The data is processed by the Machine Learning model 5. The model outputs are displayed on the page	1. Positive 2. Positive 3. Positive 4. Positive 5. Positive
A veterinarian clinic has suffered a power outage that affected the health records saved in the last hour. The cat whose health records were wiped already left the clinic. The doctor needs to create the health record again. They do not remember the cats breed but they used the Orion App to determine the breed. The Vet needs to get the cat breed again but does not have a photo of the cat saved.	1. Vet opens the Orion App 2. Vet presses the History tab 3. Vet presses on the latest entry in the History table 4. Vet can now access the classification results	1. The app launches 2. The History tab view loads and fetches data 3. The History item view loads and fetches data 4. The fetched item data is displayed	1. Positive 2. Positive 3. Positive 4. Positive
Kimmy is interested in getting a Bengal cat but does not know much about the breed and wants to learn more. Kimmy needs to access information about Bengal cats.	1. Kimmy opens the Orion app 2. Kimmy presses on the Encyclopedia tab 3. Kimmy finds the Bengal cat by scrolling through the table 4. Kimmy presses on the Bengal cat picture 5. Kimmy presses Learn More at the bottom of the Bengal cat page 6. Kimmy can now access information about the Bengal cat	1. The app launches 2. The Encyclopedia tab view loads and fetches data 3. The table is presented and allows for scrolling 4. The Bengal Encyclopedia page opens and fetches data 5. The external URL opens leading to a website with more information 6. External URL loads	1. Positive 2. Positive 3. Positive 4. Positive 5. Positive 6. Positive

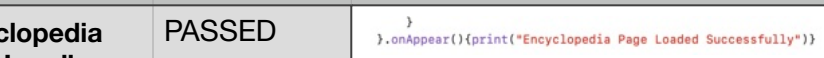
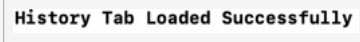
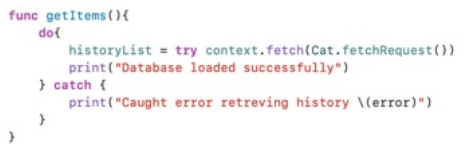
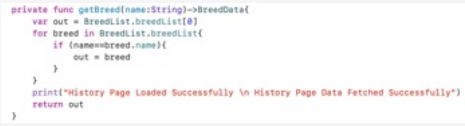
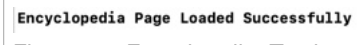
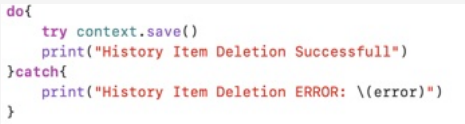
User Story	Task Steps	Desired Outcome	Step Outcome
Cynthia just found a homeless cat that she decided to take in. She wants to learn more about the cat as she build a bond with them. Cynthia wants to see if the cat is of a certain breed.	1. Cynthia opens the Orion App 2. Cynthia presses Select Photo 3. Cynthia presses Camera in the input menu 4. Cynthia takes photo of the cat and submits it 5. Cynthia looks at the classification results 6. Cynthia presses at the each of the result to read more about the breed	1. The app launches 2. The Photo Picker menu opens 3. The camera app launches 4. The data is processed by the Machine Learning model 5. The model outputs are displayed on the page 6. The corresponding Encyclopedia items are loaded and displayed	1. Positive 2. Positive 3. Positive 4. Positive 5. Positive 6. Positive
Lisa sees a photo of an adorable cat on instagram. She wants a cat that looks exactly like the one on the photo but it is not clear what breed that cat is. Lisa wants to find out the breed of the cat.	1. Lisa saves the photo from Instagram 2. Lisa opens the Orion App 3. Lisa Presses Select Photo 4. Lisa presses Photo Library in the input menu 5. Lisa selects the saved photo from Instagram and submits it 6. Lisa looks at the classification results and determines what breed the cat is	1. The photo is saved to Lisas Photo Library 2. The app launches 3. The Photo Picker menu opens 4. The Photo Library launches 5. The data is processed by the Machine Learning model 6. The model outputs are displayed on the page	1. Positive 2. Positive 3. Positive 4. Positive 5. Positive 6. Positive
Kyle uses the Orion app a lot but she is currently running out of storage on her phone. The photos stored by the Orion app are taking a lot of space. Kyle needs to delete some of the photos stored by the Orion app.	1. Kyle opens the Orion app 2. Kyle pressed History 3. Kyle scrolls through the History table to find a photo she wants to delete 4. Kyle swipes to the left and presses delete to get rid of the photo 5. Kyle repeats the process for every photo she wants gone	1. The app launches 2. The History tab view loads and fetches data 3. The History tab view presents the table and allows to scroll through it 4. The photo is deleted from the database 5. Each photo deleted by Kyle is deleted from the database	1. Positive 2. Positive 3. Positive 4. Positive 5. Positive

5.1.2 Functionality Testing

The functionality testing has been conducted through a series of test messages that are printed after all the desired code has been executed. Once the test message is printed without any critical warnings the test counts as passed. This was also verified by visual inspection of correct app visual behaviours.

Test Description	Passed?	Test Message Implementation	Terminal Output
Launch Testing: Test message implemented in the ContentView struct	PASSED	 Figure 59: Functionality Testing Code(1)	 Figure 75: Functionality Testing Terminal Output(1)
Classifier Launch: Test message implemented in the ContentView struct	PASSED	 Figure 60: Functionality Testing Code(2)	 Figure 76: Functionality Testing Terminal Output(2)
Classifier Photo Picker: Test messages implemented in the Classifier struct inside the PhotoPicker	PASSED	 Figure 61: Functionality Testing Code(3)	 Figure 77: Functionality Testing Terminal Output(3)
Classifier Photo Binary Classification: Test messages implemented in the Classifier struct in the detectCat() function	PASSED	 Figure 62: Functionality Testing Code(4)	 Figure 78: Functionality Testing Terminal Output(4)

Test Description	Passed?	Test Message Implementation	Terminal Output
Classifier Photo Classification: Test messages implemented in the Classifier struct in the classifyCat() function	PASSED	 Figure 63: Functionality Testing Code(5)	 Figure 79: Functionality Testing Terminal Output(5)
Classifier Photo Compression & Saving: Test messages implemented in the Classifier struct in the classifyCat() function	PASSED	 Figure 64: Functionality Testing Code(6)	 Figure 80: Functionality Testing Terminal Output(6)
Classifier Label Display: Test message implemented in the Classifier struct in the classifyCat() function	PASSED	 Figure 65: Functionality Testing Code(7)	 Figure 81: Functionality Testing Terminal Output(7)
Classifier Encyclopedia Links: Test messages implemented in the BreedView struct at the end of the body view	PASSED	 Figure 66: Functionality Testing Code(8)	 Figure 82: Functionality Testing Terminal Output(8)
Classifier Page Clearing: Test messages implemented in the Classifier struct in the body view	PASSED	 Figure 67: Functionality Testing Code(9)	 Figure 83: Functionality Testing Terminal Output(9)
Encyclopedia Tab Loading: Test message implemented in the Encyclopedia struct	PASSED	 Figure 68: Functionality Testing Code(10)	 Figure 84: Functionality Testing Terminal Output(10)

Test Description	Passed?	Test Message Implementation	Terminal Output
Encyclopedia Page Loading: Test messages implemented in the BreedView struct at the end of the body view	PASSED	 Figure 69: Functionality Testing Code(11)	 Figure 85: Functionality Testing Terminal Output(11)
History Tab Loading: Test message implemented in the History struct	PASSED	 Figure 70: Functionality Testing Code(12)	 Figure 86: Functionality Testing Terminal Output(12)
History Data Fetching: Test message implemented in the History struct in the getItems() function	PASSED	 Figure 71: Functionality Testing Code(13)	 Figure 87: Functionality Testing Terminal Output(13)
History Page Loading: Test message implemented in the History struct in the getBreed() function	PASSED	 Figure 72: Functionality Testing Code(14)	 Figure 88: Functionality Testing Terminal Output(14)
History Encyclopedia Loading: Test messages implemented in the BreedView struct at the end of the body view	PASSED	 Figure 73: Functionality Testing Code(15)	 Figure 89: Functionality Testing Terminal Output(15)
History Deleting Entries: Test message implemented in the History struct in the removeEntry() function	PASSED	 Figure 74: Functionality Testing Code(16)	 Figure 90: Functionality Testing Terminal Output(16)

Terminal Output

```
APP LOADED SUCCESSFULLY
Encyclopedia Tab Loaded Successfully
History Tab Loaded Successfully
NAVIGATION BAR LOADED SUCCESFULLY
Classifier Tab Loaded Successfully
Photo Picker - Library Input Loaded Successfully
Image Loaded Successfully
2023-04-08 14:14:21.329027+0200 Orion[63314:30139806] Metal API Validation Enabled
Binary Model Loaded Successfuly
Binary Prediction Successfull
Classification Model Loaded Successfuly
Classification Prediction Successfull
Before Image size in mb:2.008042
Compression success!
 Before Image size in mb:2.008042
After Image size in mb:1.155014
Photo Save Success!
Labels Updated Successfully!
Photo Picker - Camera Input Loaded Successfully
2023-04-08 14:14:32.507066+0200 Orion[63314:30139806] [Camera] Attempted to change to mode Portrait with an unsupported device
    (BackDual). Auto device for both positions unsupported, returning Auto device for same position anyway (BackAuto).
2023-04-08 14:14:32.507205+0200 Orion[63314:30139806] [Camera] Attempted to change to mode Portrait with an unsupported device
    (BackDual). Auto device for both positions unsupported, returning Auto device for same position anyway (BackAuto).
2023-04-08 14:14:32.508391+0200 Orion[63314:30139806] [Camera] Attempted to change to mode Portrait with an unsupported device
    (BackAuto). Auto device for both positions unsupported, returning Auto device for same position anyway (BackAuto).
2023-04-08 14:14:32.508450+0200 Orion[63314:30139806] [Camera] Attempted to change to mode Portrait with an unsupported device
    (BackDual). Auto device for both positions unsupported, returning Auto device for same position anyway (BackAuto).
```

Figure 91: Functionality Testing Terminal Output(1)

```
Image Loaded Successfully
Binary Model Loaded Successfuly
Binary Prediction Successfull
Classification Model Loaded Successfuly
Classification Prediction Successfull
Before Image size in mb:11.973117
Compression success!
 Before Image size in mb:11.973117
After Image size in mb:2.16297
Photo Save Success!
Labels Updated Successfully!
Classifier Tab Loaded Successfully
Page Cleared successfully!
Image Loaded Successfully
Encyclopedia Page Loaded Successfully
Database loaded successfully
History Page Loaded Successfully
 History Page Data Fetched Successfully
History Page Loaded Successfully
 History Page Data Fetched Successfully
History Page Loaded Successfully
 History Page Data Fetched Successfully
Encyclopedia Page Loaded Successfully
History Item Deletion Successfull
```

Figure 92: Functionality Testing Terminal Output(2)

5.1.3 User Testing

2 rounds of user tests have been conducted over a 2 week period with 3 users participating each round.
This is how the app looked at the point of the first round of testing:

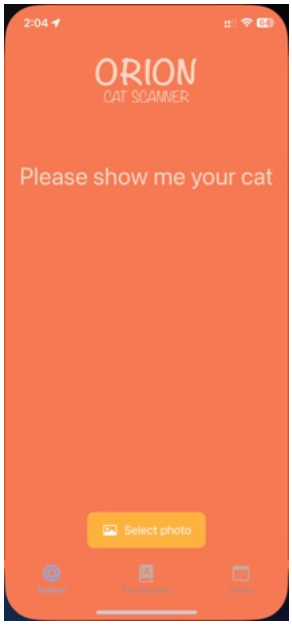


Figure 93: User Testing Before App (1)

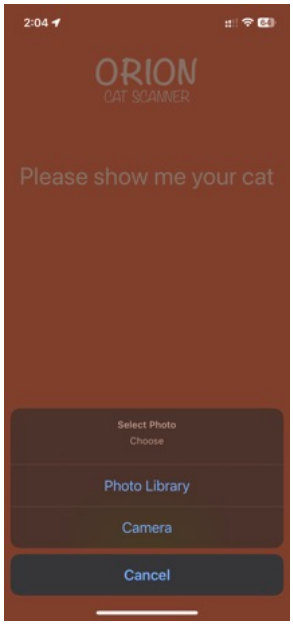


Figure 94: User Testing Before App(2)

Figure 95: User Testing Before App(3)

Figure 96: User Testing Before App(4)

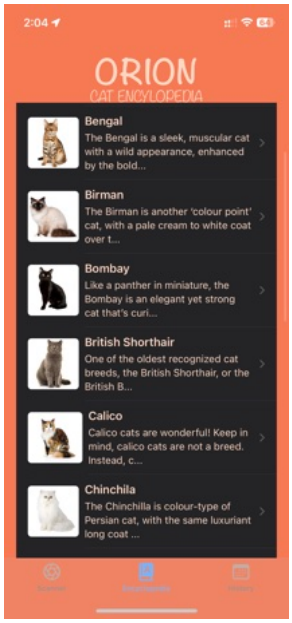


Figure 97: User Testing Before App(5)

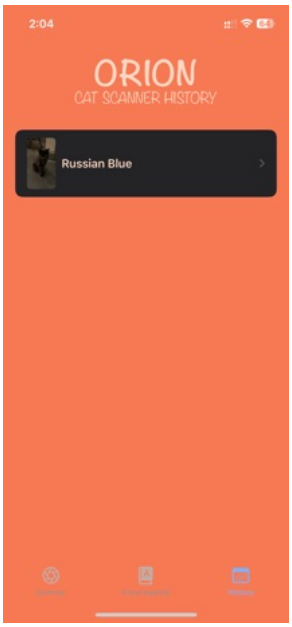


Figure 98: User Testing Before App(6)

Round 1 Results

Question	User 1	User 2	User 3
Q1: On a scale of 1 to 10 how would you rate the app interface?	7/10	7/10	8/10
Q2: What part of the app was your favorite and why?	Encyclopedia & “Learn More” button linked to Purina website with loads of details	Encyclopedia with cute cat photos	Encyclopedia as it is interesting and has a lot of adorable photos
Q3: What part of the app was your least favorite and why?	The tab labels are too small and colors are too hard to read.	History page because of the color clashes, sorting oldest to newest and bad text formatting.	History sorting by oldest to newest. The tab buttons are not very readable.
Q4: What would you change in the app if anything and why?	Darker colors for tabs would be better and bigger font. If the app could tell that the photo is not a cat would be great. It assigns Calico to every non cat currently.	Add more space between the heading and the content. Breed Details button could be more obvious. Change the starting text in the scanner page.	I would add slide left to go back function.
Q5: Is the color scheme of the app clear to read? If not please highlight areas that are problematic.	The font should be darker color and bolder.	Yellow button on top of a orange background is not great to read. The highlights on tabs are blue and tables are black so there is a lot of color clashing. If the yellow button matched the highlighted tab color blue it would be more appealing.	Previously mentioned tab buttons. Light blue colors would ideally be more contrasted with the background.
Q6: Is the information presented by the application detailed enough?	If the scanner would name the percentages matched with all top 3 instead of just #1 & time stamp at the history screen so I can tell the order of the photos. The history screen could store more information about the classification.	The history screen should have time stamps. Percentages matching the #2 #3 labels.	Yes.
Q7: Does the wording throughout the app clearly indicate usage steps?	Change the “show me” In the scanner page to something else.	Wording is okay.	Yes.
Q8: What would you change about the Scanner tab(if not indicated before)?	Breed Details does not have a button shape which would be very helpful. Change the wording of the result.	No other changes than previously discussed.	Nothing
Q9: What would you change about the Encyclopedia tab(if not indicated before)?	Cat icons are a bit small. Maybe change the background of the cats. More contrast.	The spacing between the header and the table.	Nothing

Question	User 1	User 2	User 3
Q10: What would you change about the History tab(if not indicated before)?	Photo size to be unified. Data stamps. Text aligned.	Delete entries.	Sort new to old and present scan date.
Notes	This user has a visual impairment. User Note: Could the dataset be expanded?	User is dyslexic. User note: Chinchilla breed has a typo.	—
Indicated changes:	-History screen should present time stamps -History screen should store more information -Tab labels need to be bigger and a different color -Add cat detection -Fonts should be darker and bolder. -Scanner should show all top 3 match percentages -“Show Me” should be changed to something else(Scanner page) -Breed Details link should have a button shape -Cat images in the encyclopedia could be a bit bigger & contrasted -Photos in the history screen should be all the same ratio and the text should be aligned	-Tab colors are clashing with the app color scheme -Color clashes -Text formatting -History page should sort latest to oldest -History page should show a time stamp -Delete button for history page -More space between header and main content -Breed Details should be more visible as a button -Text in the scanner page should be changed -Yellow button on an orange background is not very readable -The match percentages should be shown for all 3 matches	-History should sort newest to oldest -Tab buttons should be easier to read -More color contrast with tab buttons -Swipe to go back

Following changes have been implemented based on user feedback:

-General Changes

-the headers have an additional space added to the bottom to allow for more space between the text and content

-it is now possible to swipe to go back from any button page

-Tab labels visibility

-the tab icons were made bigger and the color has been changed to match all other app fonts

-the tab labels were made bigger and the font has been changed to match all other font colours

-Cat detection

-a machine learning model was implemented that classifies wether a photo contains a cat or not

-Fonts

-fonts all over the app have been adjusted to be the same color and be more visible

-Scanner page

-the scanner page now presents a different opening message : “Upload Photo to Detect Cat Breeds” instead of “Please Show Me Your Cat”

-the Select Photo button has been adjusted, the color matches all other fonts and it was made more pronounced
-the photo picker text has been changed to “Select Photo” “From”
-after uploading a photo the scanner page now displays “This cat shares the features of” if the ML model detects a cat or “This is not a cat but if it was it would be” if it does not detect a cat
-three labels are now presented and they are all buttons. They present an encyclopaedia icon, awarded label and matching percentage. Upon pressing the button the matching encyclopaedia page is opened.

-History page

-the table in the history screen was made to sort newest to oldest
-the table in the history screen was made to present the photo icon and time stamp
-the history entries themselves now store all 3 labels with match rates
-the history screen presents the saved photo
-the text is now aligned in the table
-the photos are now aligned in the table
-the entries now can be deleted by swiping to the right on the entry and pressing delete

How the app looked at the time of Round 2 testing after implementing the aforementioned changes:

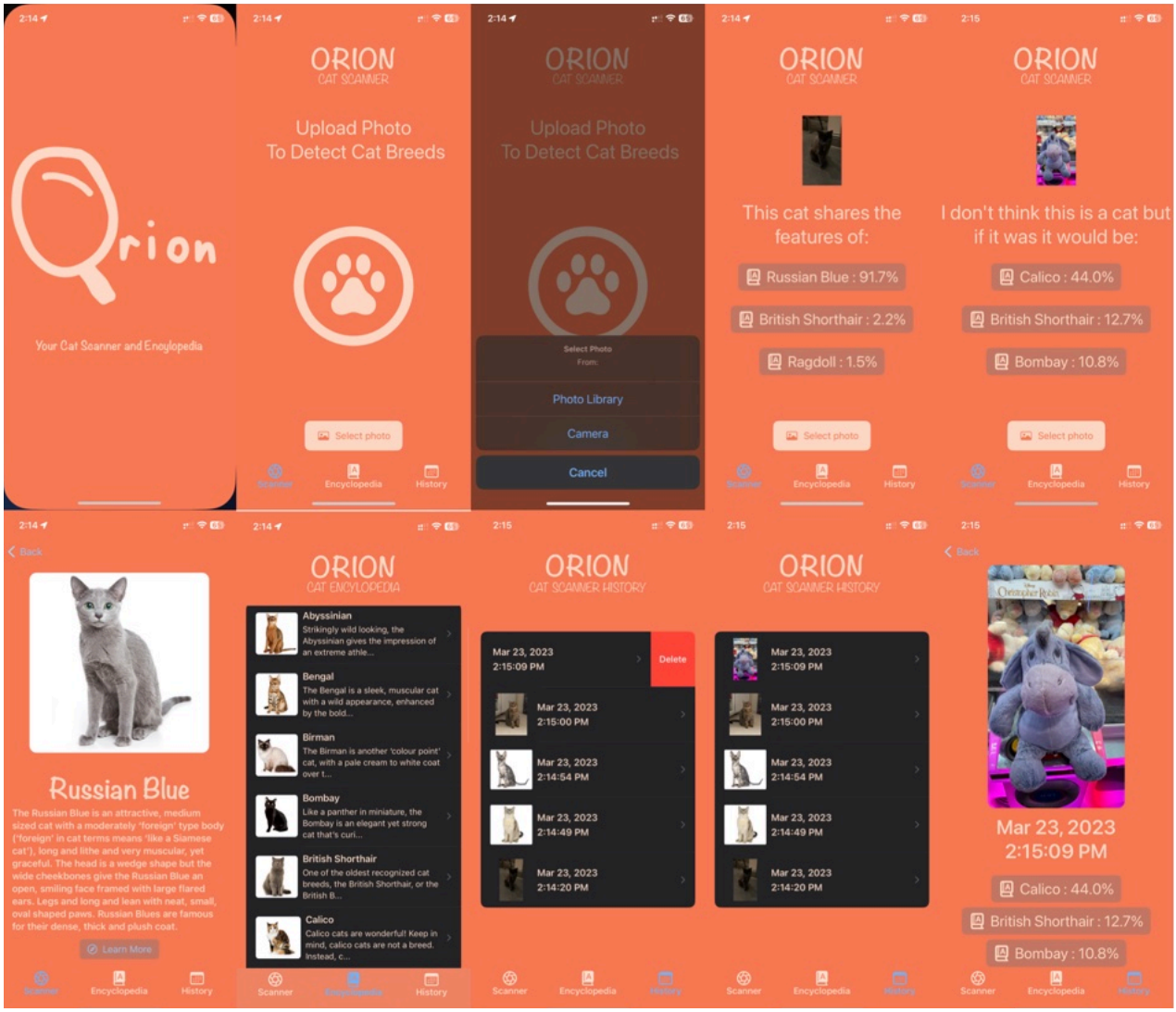


Figure 99: User Testing App Mid Testing

Round 2 Results

Question	User 1	User 2	User 3
Q1: On a scale of 1 to 10 how would you rate the app interface?	9/10	9/10	9/10
Q2: What part of the app was your favorite and why?	Same as before.	Same as before.	Same as before.
Q3: What part of the app was your least favorite and why?	None	None	None
Q4: What would you change in the app if anything and why?	Paw seems like a button. Make header bold.	Deleting history is not obvious - android user	Make the screen with the paw print home screen that you can go back to. Non-iPhone user - swipe to delete not obvious.
Q5: Is the color scheme of the app clear to read? If not please highlight areas that are problematic.	Paw should not have a circle. Pictures should be same ratio. Button should have a 3d effect to be more obvious.	Yes	Yes
Q6: Is the information presented by the application detailed enough?	Yes much better now.	Yes	Yes
Q7: Does the wording throughout the app clearly indicate usage steps?	Yes	Yes	Yes
Q8: What would you change about the Scanner tab(if not indicated before)?	Get rid of the circle around the paw	No	Header bold so it is easier to notice.
Q9: What would you change about the Encyclopedia tab(if not indicated before)?	Get rid of the circle around the paw	No	No
Q10: What would you change about the History tab(if not indicated before)?	Pictures in the same ratio.	Delete function more obvious	Header bold so it is easier to notice.
Notes	—	—	Pawprint looks like a button.
Indicated changes:	-Paw icon should not have a circle -Header should be bold -History screen photo icons should be same ratios	-Swipe to delete is not obvious	-Make the starting screen a Home Screen that you can go back to -Swipe to delete is not obvious -Bolder header -Change the paw print icon so it does not look like a button

Following changes have been implemented based on user feedback:

- -General Changes
  - -Paw icon does not have a circle around it now
  - -Headers have been made more visible
- -Scanner Page
  - -The welcome page is now a home page that you can go back to by pressing "Clear Page"

The swipe to delete function will be explained in the user manual.

This is how the app presented after implementing aforementioned changes:

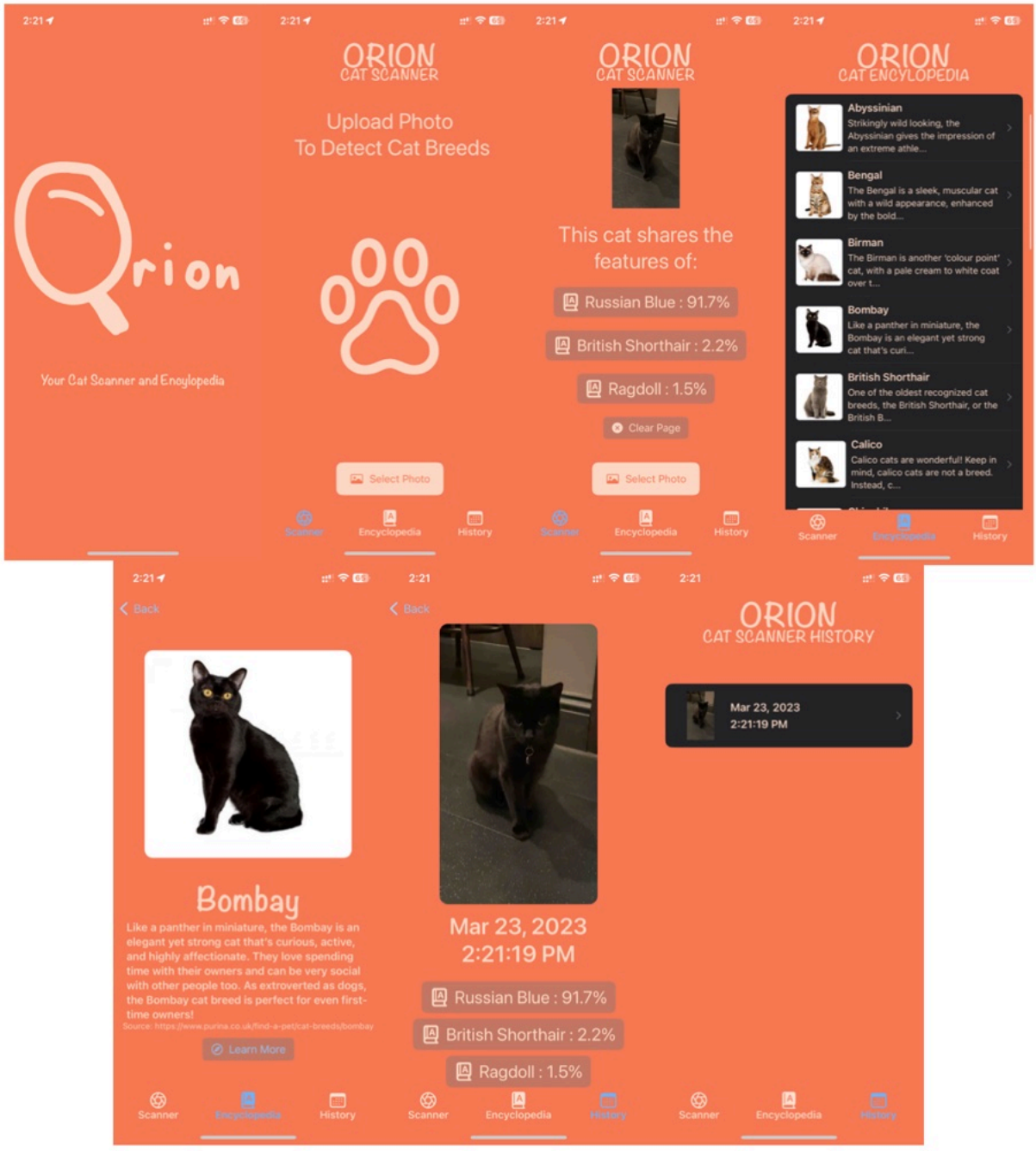


Figure 100: User Testing Final App State

User Testing Summary

Throughout the conducted user testing a plethora of changes desired by users have been implemented. The changes have made the app interface much clearer and easier to read. The additional functionality that was added were key functions to ensure user comfort. During the first round of testing the users all had very similar feedback concerning interface clarity. 2 out of the 3 users are people with visual impairments. The color clashes and hard to read labels were a big issue. The wording throughout the app was thought to not always be very appropriate. Users were not happy with the history page sorting history entries by oldest to newest and no ability to delete entires. The tests indicated that not enough detailed information was presented and stored by the app. This feedback has been taken to improve the app. Over 23 key changes resulted from the first round of the user testing.

The second part of user tests brought much less call for changes from the users which meant that implemented changes brought a big improvement to the app. This turn there was only a single request for new functionality - adding a home screen. Other changes were very minor visual improvements. When it comes to the swipe to delete not being obvious - both users reporting that feedback are native Android users where that is not common. The third user that is a native iOS user did not report that as an issue. This will not be tackled by any software changes and will simply be explained in the app manual.

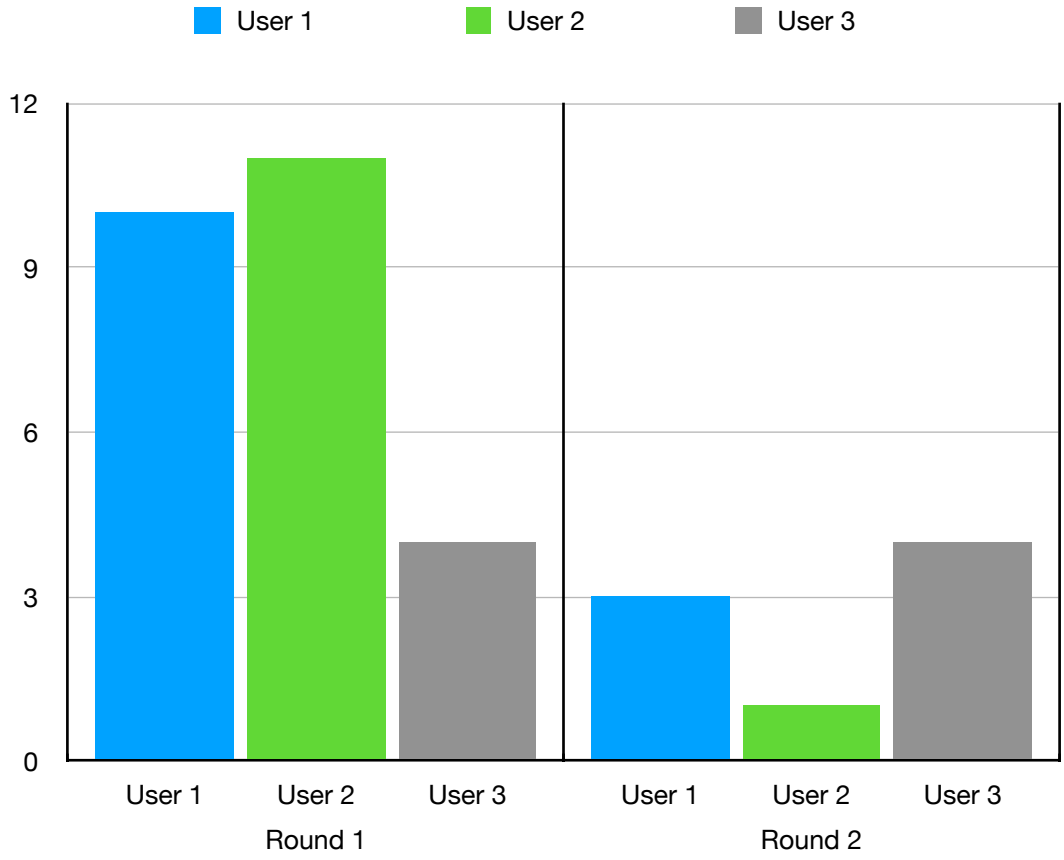


Figure 101: User Testing Changes Desired By Users Graph

User App Rating Per Round

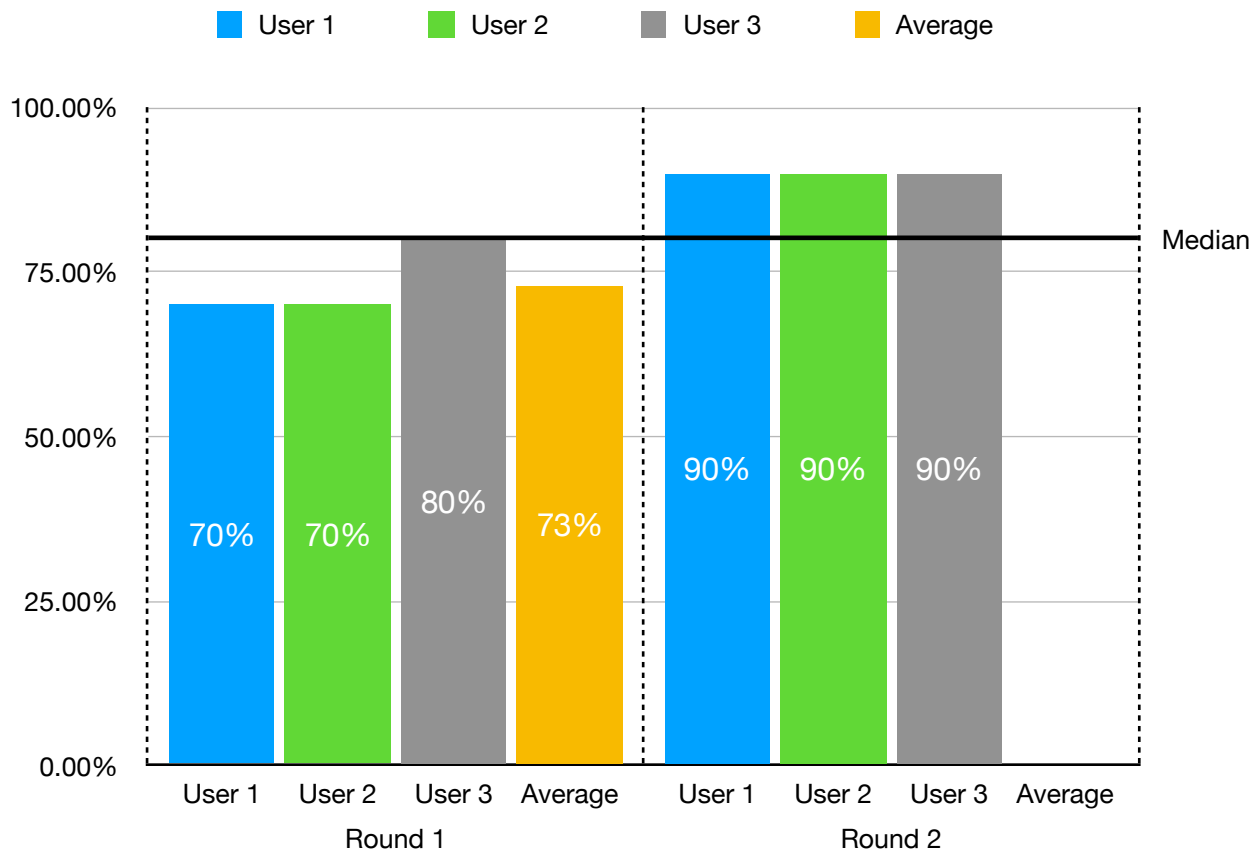


Figure 102: User Testing App Rating Graph

The biggest changes when comparing the app version before any tests conduced versus the version after all tests were concluded are shown on the next page (before, after):

1. Home Screen

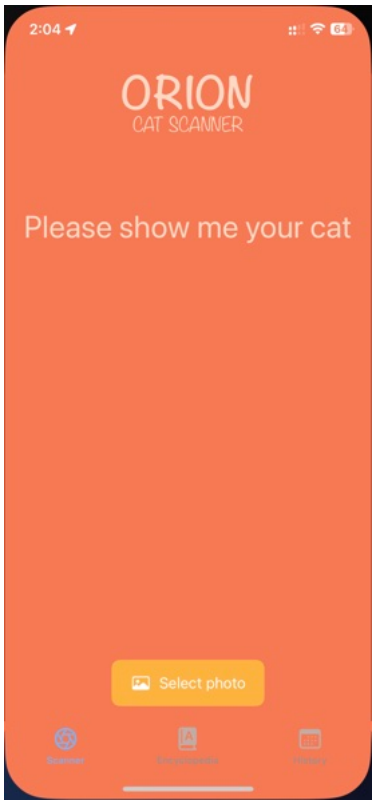


Figure 103: User Testing Home Screen Before

Figure 104: User Testing Home Screen After

2. Result Screen

Figure 105: User Testing Result Screen Before

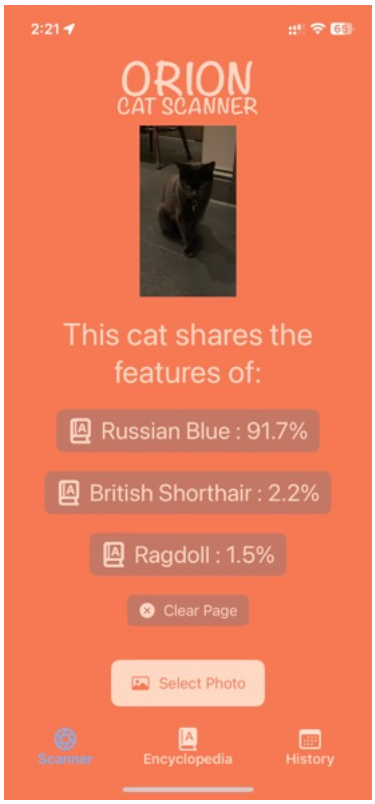


Figure 106: User Testing Result Screen After

3. History Item Screen

Figure 107: User Testing History Item Before

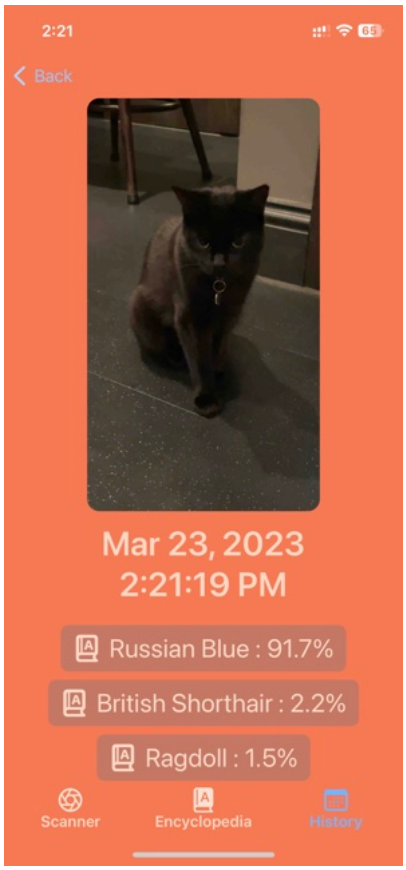


Figure 108: User Testing History Item After

5.1.4 Market Comparison

On the iOS App Store I was able to find 4 apps that offer similar functionality to the Orion: Cat Scanner & Encyclopaedia app.

1. Cats Pedia: Breed identifier

Functionality	CatsPedia	Orion: Cat Scanner & Encyclopedia
Cat Breed Encyclopedia	• Consists of 95 breeds • Some breeds are missing data • You can filter by traits and name • You can favorite breeds • Presents detailed information • Different layouts available	• Consists of 20 breeds • No missing data • No filtering • No favorite function • Presents generalized information with external links to learn more • Single layout
History Page	• Has history page • Swipe to delete • Needs internet connection to load • Presents small icon of the saved photo • Saves top 3 results • Saves date & time	• Has history page • Swipe to delete • Works offline • Presents small icon of the saved photo as well as a big photo • Saves top 3 results • Saves date & time
Classification Page	• Needs internet connection to load • Passed data outside of your device • Takes time to load results • Buggy - freezes app • Low accuracy on tested photos	• Works fully offline • Keeps all the data on your device • Results instant • Bug free • High accuracy on tested photos
Others	• Ads • Additional Functionality: - Settings • Buggy photo picker	• Ad free • No Additional Functionality • Bug free photo picker

2. Cat Scanner Breed Identifier

Functionality	Cat Scanner	Orion: Cat Scanner & Encyclopedia
Cat Breed Encyclopedia	• Consists of 59 breeds(includes wild cats such as Lions) • You can filter by traits and name • Presents detailed information	• Consists of 20 breeds • No filtering • Presents generalized information with external links to learn more

Functionality	Cat Scanner	Orion: Cat Scanner & Encyclopedia
History Page	• Has history page • Swipe to delete & more • Delete all history • Needs internet connection to load • Presents small icon of the saved photo as well as a big photo • Saves top 3 results • Does not save date & time	• Has history page • Swipe to delete • Delete single entires • Works offline • Presents small icon of the saved photo as well as a big photo • Saves top 3 results • Saves date & time
Classification Page	• Needs internet connection to load with limited Offline option • Passed data outside of your device • Takes time to load results • High accuracy on tested photos	• Works fully offline • Keeps all the data on your device • Results instant • Bug free • High accuracy on tested photos

Others	• Ads
	• Additional Functionality:
	  - Social Media
	  - Sharing Results
	  - Settings
		• Ad free
		• No Additional Functionality

3. Breed Recognizer

Functionality	Breed Recognizer	Orion: Cat Scanner & Encyclopedia
Cat Breed Encyclopedia	• Consists of 68 breeds • No filtering • Presents extremely limited information • No photos	• Consists of 20 breeds • No filtering • Presents generalized information with external links to learn more • Has example photo per breed
History Page	• No history page	• Has history page • Swipe to delete • Delete single entires • Works offline • Presents small icon of the saved photo as well as a big photo • Saves top 3 results • Saves date & time
Classification Page	• Needs internet connection to load • Passed data outside of your device • Takes a very long time to load results • Accuracy not checked as would not load results	• Works fully offline • Keeps all the data on your device • Results instant • Bug free • High accuracy on tested photos

Functionality	Breed Recognizer	Orion: Cat Scanner & Encyclopedia

Others	• Ads
	• Very poor experience, app is very slow and crashes a lot
	• Additional Functionality:
	  - List your pet as missing
	  - Adopt a pet
	  - Supports dogs
		• Ad free
		• Very smooth fast crash free experience
		• No Additional Functionality

4.Cat Breed Identifier: Pet Scan

Functionality	Cat Breed Identifier: Pet Scan	Orion: Cat Scanner & Encyclopedia
Cat Breed Encyclopedia	• Consists of 80 breeds • No filtering • Search by name • Presents general information • Multiple photos	• Consists of 20 breeds • No filtering • No search function • Presents generalized information with external links to learn more • Single photo per breed
History Page	• History page is locked behind a paywall so it was not tested	• Has history page • Swipe to delete • Delete single entires • Works offline • Presents small icon of the saved photo as well as a big photo • Saves top 3 results • Saves date & time
Classification Page	• Needs internet connection to load • Passed data outside of your device • Takes a long time to load results • Accuracy was average on tested photos	• Works fully offline • Keeps all the data on your device • Results instant • Bug free • High accuracy on tested photos
Others	• Ads • Additional Functionality such as offline mode and history was not tested as locked behind paywall	• Ad free • No Additional Functionality

Chapter 5.1.5 Evaluation Summary

This chapter focused on the evaluation of Orion: Cat Scanner & Encyclopaedia.
The App meets all the planned requirements. All functionality has been implemented fully and tested thoroughly.
After conducting User Tests the app was improved greatly thanks to all the changes proposed by the users. Throughout it's development the app saw tremendous growth and is now a very user friendly and fast experience.
Compared to other apps that offer similar functionality on the market Orion: Cat Scanner & Encyclopaedia performs very well. The main advantages compared to other apps were: data security that is inherent with local operations only as no data passes to a server there is no risk of data leaks, no need for network connection for the app to work allows the user to conduct the classification anywhere and it provides a much smoother experience as there is no delay for the model prediction - the results are instantaneous which no other app that was compared offered.

The evaluation results are very satisfactory proving that Orion: Cat Scanner & Encyclopaedia is a very high quality application with intuitive and fast user experience.

Chapter 5.2: Machine Learning Evaluation

5.2.1 Cat Breed Classification

The final model proved to be highly accurate as confirmed in evaluation.
During the training process the model achieved an average training accuracy of over 92% with average loss comfortably staying under 0.23.
Validation training accuracy reached over 90% multiple times during the training process with loss staying under 0.5 for the most part. The Validation variables have a much grater variability. The graphs from a 500 epoch long training are presented below:

The additional evaluation metrics measured were implemented.
The other metrics measured were a test accuracy done on a dataset of 675 images, F1 score, ROC-AUC curve, Precision and Recall. Results:

```
Test accuracy is :  94.23999786376953 %
F1 score is 0.9472836163230793
ROC- AUC score is 0.9988429967866688
Precision is:  0.9433090515707214
Recall is:  0.943981685229485
```

A confusion matrix produced a very satisfying result:

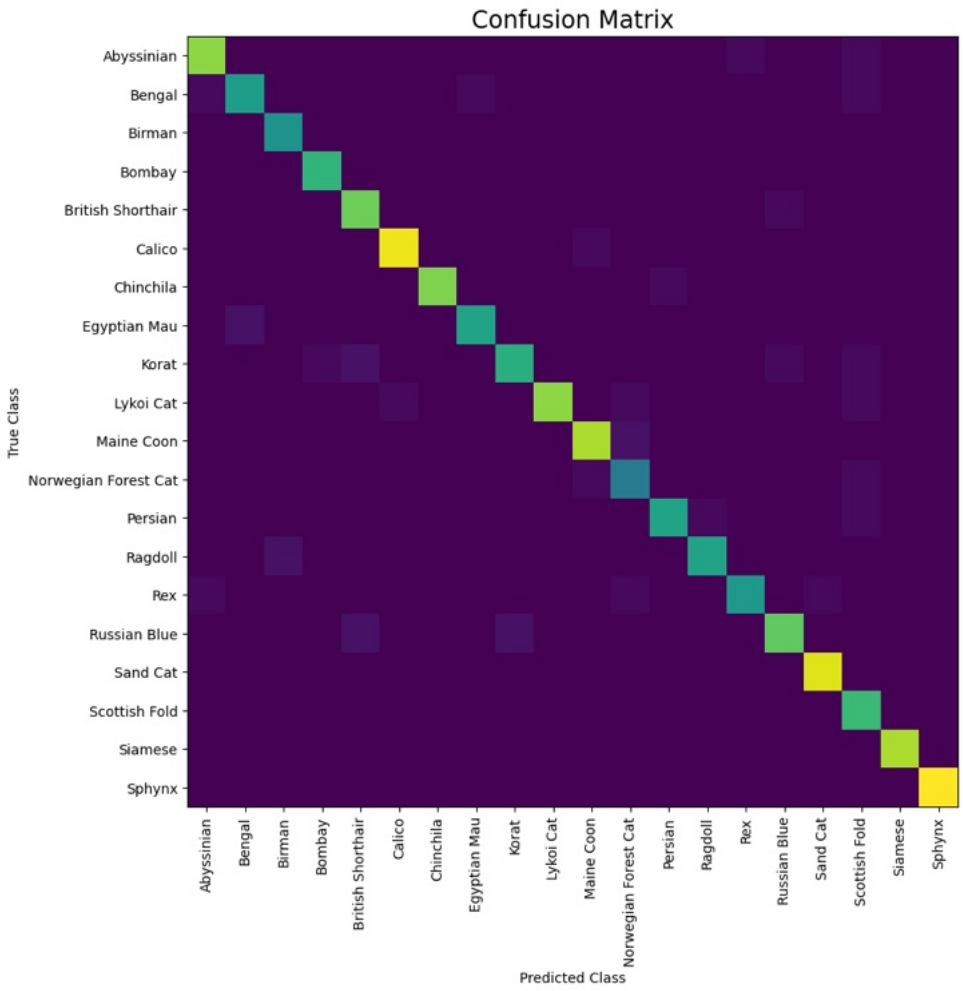


Figure 109: Machine Learning Breed Classification Confusion Matrix

5.2.2 Cat Detection Through Binary Classification

The model used for detection was proven to be highly accurate which was confirmed through multiple accuracy metrics measured.
The test metrics used and their results were as follows:

```
Test accuracy is :  99.46646094322205 %
F1 score is 0.9885775467139927
ROC- AUC score is 0.9888515637278014
Precision is:  0.9870634922576683
Recall is:  0.9888515637278014
```

With a Confusion Matrix:

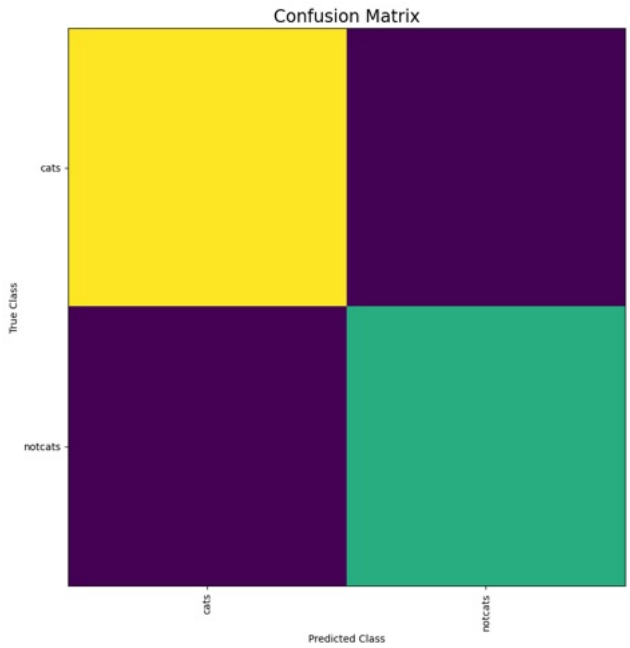


Figure 110: Machine Learning Cat Detection Through Binary Classification Confusion Matrix

Chapter 6: Project Management

6.1 Project Development Plan

The projects first plan was created during challenge week in the Autumn of 2022. The main tool for planning and management was Jira with Gitlab being used for storing and updating the project.
A Gantt-Chart was produced using Google Sheets to track progress as well.

Three main milestones were identified. The milestones were identified based on the amount of work and the stage of work to be done before a certain date. The end of challenge week was the first milestone. It has a strict timeline and clearly laid out objectives to accomplish. Each milestone was represented as a Jira Epic with Stories representing complex objectives and subtasks breaking down the complex objectives into simple tasks accomplishable in a span of a week to few weeks max.

Each Week there would be a Sub-Task titled “Week XX Summary”. This task was used to summarise the weekly progress and make meetings with the project supervisor easy to navigate through.

Challenge Week

During Challenge Week 6 Stories were identified with 7 sub tasks. The duration of the Challenge Week was just 5 days therefore the Issue Complexity is very low.

								Challenge Week
Task ID	Linked Issue	TASK TITLE	TASK TYPE	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE	Week 1
C301092-2	N/A	Challenge Week Objectives	Epic	Week 1	Week 1	5 days	100%	
C301092-3	C301092-2	Create a Risk Assesement	Story	Week 1	Week 1	5 days	100%	
C301092-4	C301092-2	Background Reading Summary	Story	Week 1	Week 1	5 days	100%	
C301092-5	C301092-2	Technical Deliverables	Story	Week 1	Week 1	5 days	100%	
C301092-8	C301092-2	Project Plan	Story	Week 1	Week 1	5 days	100%	
C301092-9	C301092-2	Summer Preparation Record	Story	Week 1	Week 1	5 days	100%	
C301092-41	C301092-2	Prepare a Presentation	Story	Week 1	Week 1	5 days	100%	
C301092-42	C301092-9	GUI Sketches	Sub-Task	Week 1	Week 1	5 days	100%	
C301092-43	C301092-9	Background Reading	Sub-Task	Week 1	Week 1	5 days	100%	
C301092-6	C301092-5	Algorithm	Sub-Task	Week 1	Week 1	5 days	100%	
C301092-7	C301092-5	Create a GUI	Sub-Task	Week 1	Week 1	5 days	100%	
C301092-13	C301092-11	Basic Cat Detection	Sub-Task	Week 1	Week 1	5 days	100%	
C301092-18	C301092-5	Simple GUI	Sub-Task	Week 1	Week 1	5 days	100%	
C301092-22	C301092-2	Functionality Implementation: Image Processing	Sub-Task	Week 1	Week 1	5 days	100%	

Figure 111: Gantt-Chart Challenge Week

Road to Week 11

The Autumn term ended with the Interim Oral Interview. This whole milestone was put into an Epic fittingly called “Road to Week 11”. This epic aimed to accomplish three major goals: develop a working beta version of the app with fully implemented functionality, train a Machine Learning model capable of classifying cat breeds with high accuracy. This culminated with the Interim Oral Interview. The hardest challenge of planning out Complex tasks to be done over a large period of time is approximating how much time each issue would take. This would prove to be a big problem during this Epic causing multiple planned tasks to be postponed to the next Epic.

These being additional app functionality(Encyclopaedia and History) and achieving high model accuracy. They were moved to the next Epic.

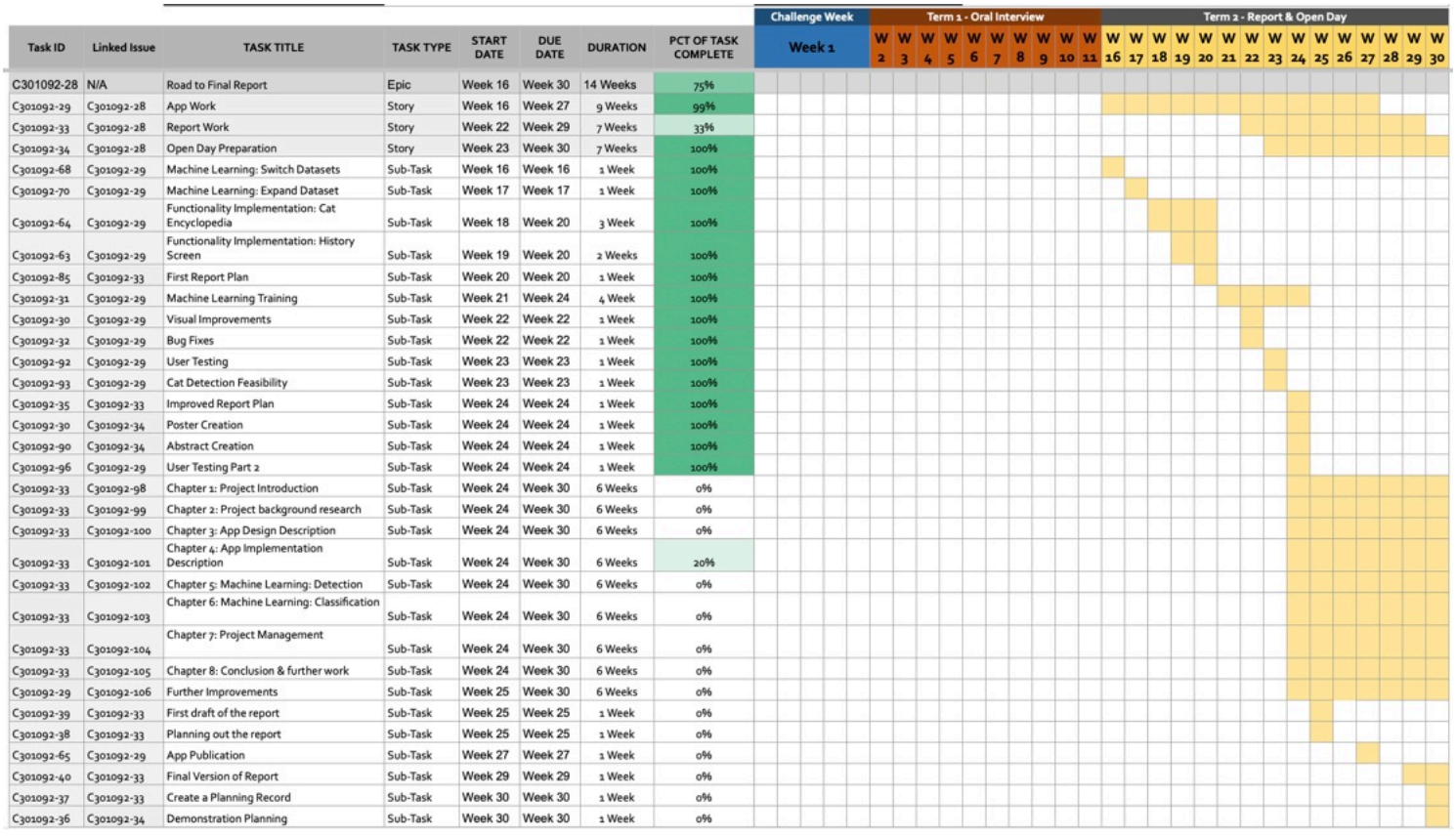
As seen below the tasks that were failed to be accomplished have 0% marked to them.

Task ID	Linked Issue	TASK TITLE	TASK TYPE	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
C301092-10	N/A	Road to Week 11	Epic	Week 2	Week 11	9 Weeks	100%
C301092-11	C301092-10	Machine Learning	Story	Week 2	Week 11	9 Weeks	100%
C301092-12	C301092-10	App Development	Story	Week 2	Week 11	9 Weeks	100%
C301092-14	C301092-11	Advanced Cat Detection	Sub-Task	Week 4	Week 4	1 Week	100%
C301092-19	C301092-12	GUI Improvements Week 4	Sub-Task	Week 4	Week 4	1 Week	100%
C301092-15	C301092-11	Simple Cat Breed Classification	Sub-Task	Week 5	Week 5	1 Week	100%
C301092-23	C301092-12	Functionality Implementation: Breed Search Screen	Sub-Task	Week 6	Week 6	1 Week	100%
C301092-20	C301092-12	GUI Improvements Week 6	Sub-Task	Week 6	Week 6	1 Week	100%
C301092-16	C301092-11	Interim Cat Breed Classification	Sub-Task	Week 7	Week 7	1 Week	100%
C301092-24	C301092-12	Functionality Implementation: Scan History Screen	Sub-Task	Week 8	Week 8	1 Week	100%
C301092-52	C301092-11	TensorFlow to CoreML	Sub-Task	Week 8	Week 8	1 Week	100%
C301092-17	C301092-11	Advanced Cat Breed Classification	Sub-Task	Week 9	Week 9	1 Week	100%
C301092-20	C301092-12	GUI Improvements Week 9	Sub-Task	Week 9	Week 9	1 Week	100%
C301092-25	C301092-12	Functionality Implementation: Breed Search	Sub-Task	Week 9	Week 9	1 Week	0%
C301092-26	C301092-12	Functionality Implementation: Scan History	Sub-Task	Week 9	Week 9	1 Week	0%
C301092-21	C301092-12	GUI Improvements Week 10	Sub-Task	Week 10	Week 10	1 Week	100%
C301092-27	C301092-12	Cat Breed Data Gathering & Database Assembly	Sub-Task	Week 10	Week 10	1 Week	0%

Figure 112: Gantt-Chart Road to Week 11

Road to Final Report

This milestone is the final of all three. It aims to achieve three major goals: finish the App, prepare for the open day and create the final report. The App Work Story includes the moved tasks from the previous Epic.



Task ID	Linked Issue	TASK TITLE	TASK TYPE	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
C301092-28	N/A	Road to Final Report	Epic	Week 16	Week 30	14 Weeks	75%
C301092-29	C301092-28	App Work	Story	Week 16	Week 27	9 Weeks	99%
C301092-33	C301092-28	Report Work	Story	Week 22	Week 29	7 Weeks	33%
C301092-34	C301092-28	Open Day Preparation	Story	Week 23	Week 30	7 Weeks	100%
C301092-68	C301092-29	Machine Learning: Switch Datasets	Sub-Task	Week 16	Week 16	1 Week	100%
C301092-70	C301092-29	Machine Learning: Expand Dataset	Sub-Task	Week 17	Week 17	1 Week	100%
C301092-64	C301092-29	Functionality Implementation: Cat Encyclopedia	Sub-Task	Week 18	Week 20	3 Week	100%
C301092-63	C301092-29	Functionality Implementation: History Screen	Sub-Task	Week 19	Week 20	2 Weeks	100%
C301092-85	C301092-33	First Report Plan	Sub-Task	Week 20	Week 20	1 Week	100%
C301092-31	C301092-29	Machine Learning Training	Sub-Task	Week 21	Week 24	4 Week	100%
C301092-30	C301092-29	Visual Improvements	Sub-Task	Week 22	Week 22	1 Week	100%
C301092-32	C301092-29	Bug Fixes	Sub-Task	Week 22	Week 22	1 Week	100%
C301092-92	C301092-29	User Testing	Sub-Task	Week 23	Week 23	1 Week	100%
C301092-93	C301092-29	Cat Detection Feasibility	Sub-Task	Week 23	Week 23	1 Week	100%
C301092-35	C301092-33	Improved Report Plan	Sub-Task	Week 24	Week 24	1 Week	100%
C301092-30	C301092-34	Poster Creation	Sub-Task	Week 24	Week 24	1 Week	100%
C301092-90	C301092-34	Abstract Creation	Sub-Task	Week 24	Week 24	1 Week	100%
C301092-96	C301092-29	User Testing Part 2	Sub-Task	Week 24	Week 24	1 Week	100%
C301092-33	C301092-98	Chapter 1: Project Introduction	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-99	Chapter 2: Project background research	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-100	Chapter 3: App Design Description	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-101	Chapter 4: App Implementation Description	Sub-Task	Week 24	Week 30	6 Weeks	20%
C301092-33	C301092-102	Chapter 5: Machine Learning: Detection	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-103	Chapter 6: Machine Learning: Classification	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-104	Chapter 7: Project Management	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-105	Chapter 8: Conclusion & further work	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-29	C301092-106	Further Improvements	Sub-Task	Week 25	Week 30	6 Weeks	0%
C301092-39	C301092-33	First draft of the report	Sub-Task	Week 25	Week 25	1 Week	0%
C301092-38	C301092-33	Planning out the report	Sub-Task	Week 25	Week 25	1 Week	0%
C301092-65	C301092-29	App Publication	Sub-Task	Week 27	Week 27	1 Week	0%
C301092-40	C301092-33	Final Version of Report	Sub-Task	Week 29	Week 29	1 Week	0%
C301092-37	C301092-33	Create a Planning Record	Sub-Task	Week 30	Week 30	1 Week	0%
C301092-36	C301092-34	Demonstration Planning	Sub-Task	Week 30	Week 30	1 Week	0%

Figure 113: Gantt-Chart Road To Final Report

6.2 Management Tools

As previously mentioned the tools used in planning and management of the project were: Jira, gitlab and a Google Spreadsheets Gantt-Chart. Jira being the most vital one.

Jira

The main tool to manage the project was Jira, it was used to track progress of all the goals. Jira was updated at least weekly with detailed descriptions of progress that sometimes included attachments such as screenshots. The previous subchapter reviewed the split of goals into Epics, Stories and subtasks.

Epics would be titled appropriately for the mile stone they have to achieve. They would hold a brief description of the milestone as well. Example below:

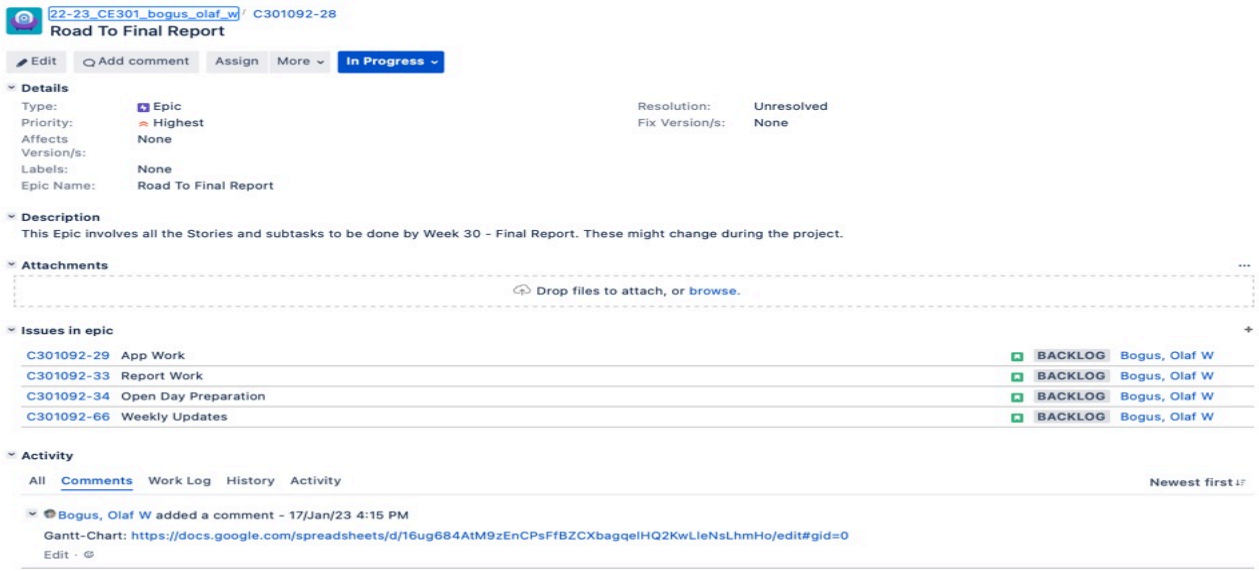


Figure 114: Jira Road to Final Report

Stories would hold titles to represent the category of sub tasks they would hold as well as the description of the overall Story goal. Example below:

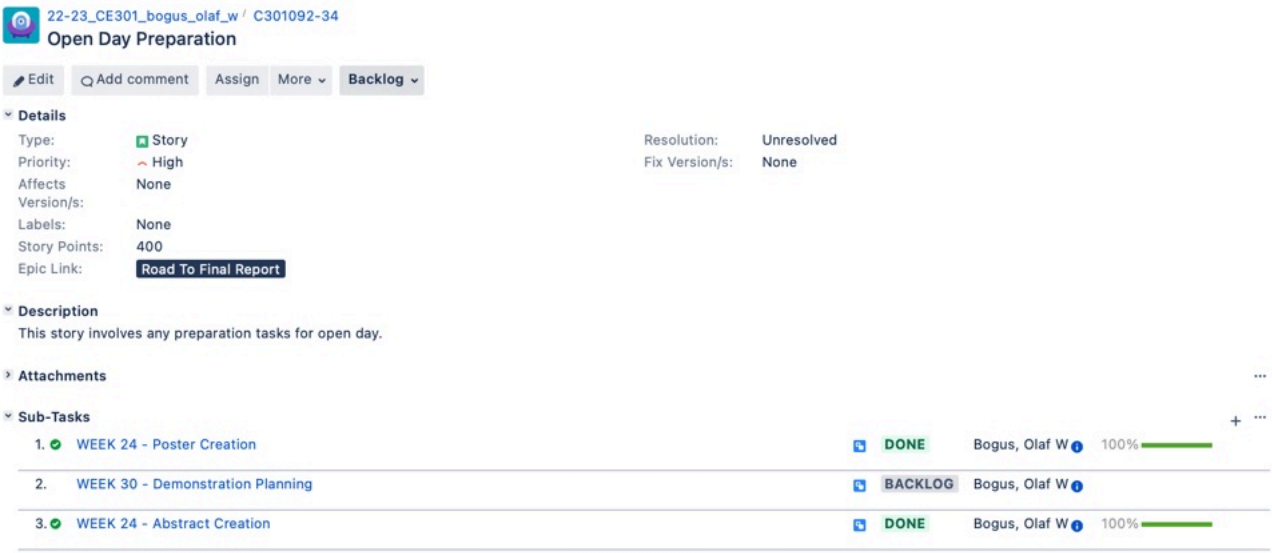


Figure 115: Jira Open Day Preparation

Sub-tasks would hold titles including the week they were meant to be done in. The detailed progress on the tasks would be described in the sub-task comments as well as attachments showing off the progress. Example below:

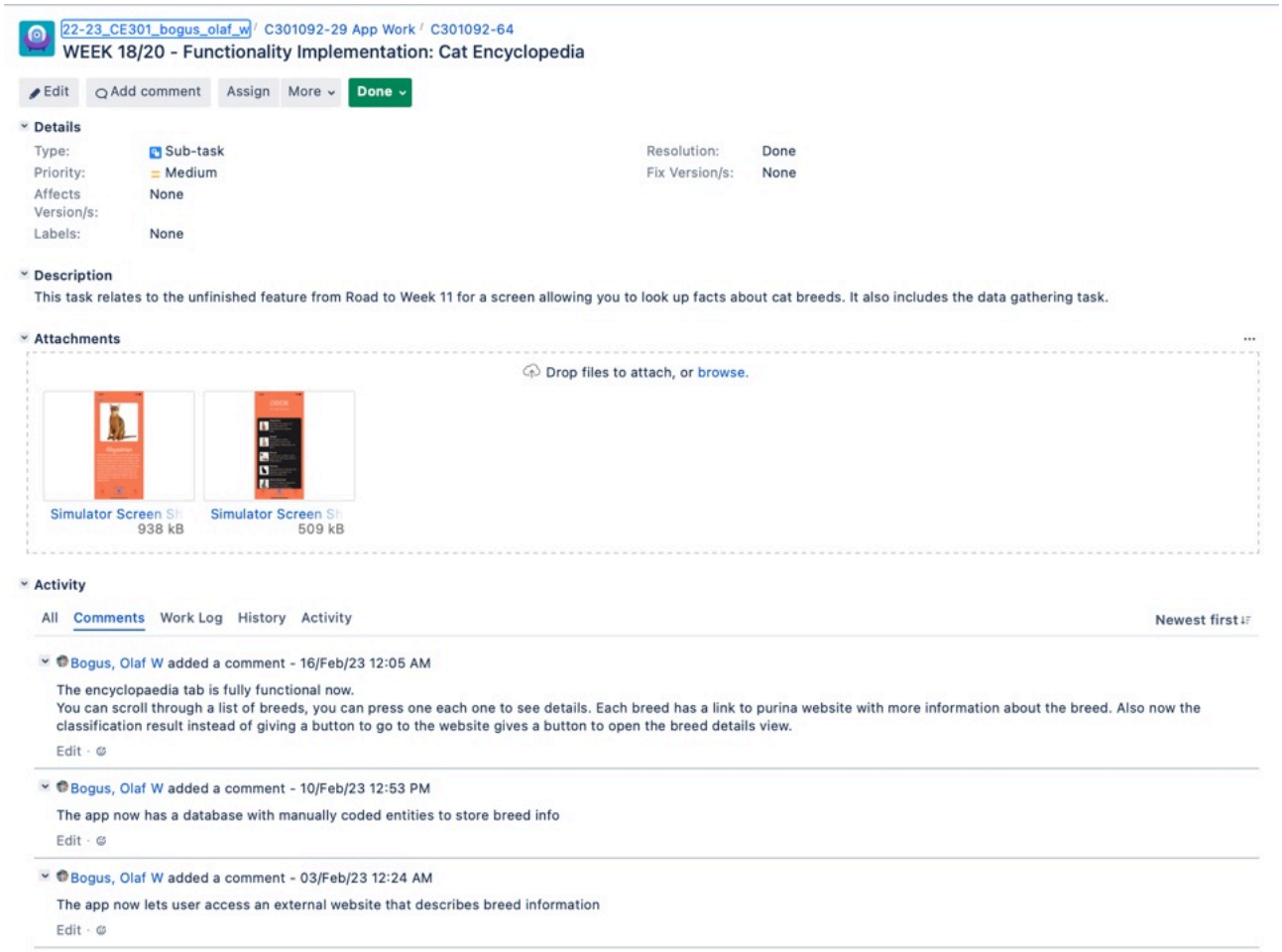


Figure 116: Jira: Functionality Implementation Encyclopaedia

Jira was also used to process feedback from the project supervisor and was the main focus of weekly meetings. This was made easier with the Weekly Summary subtasks that would hold weekly progress highlights. Example below:

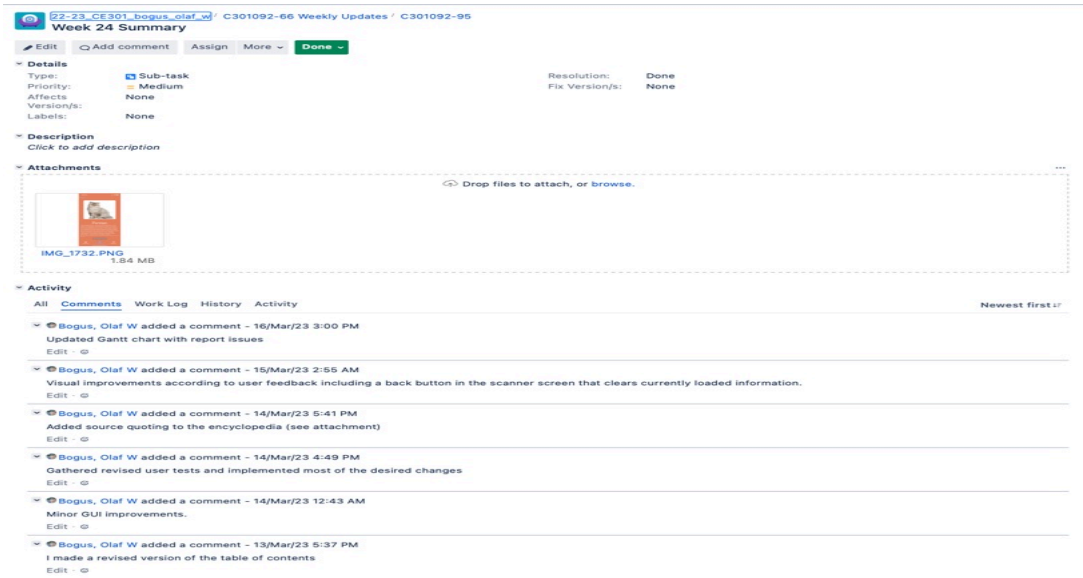


Figure 117: Jira Week 24 Summary

GitLab

This tool was used to storing and updating project files. Each commit would describe the changes included in the commit. The GitLab usage was low during the first as my unfamiliarity with Xcode software and the poor Git integration proved regular commits difficult. Due to time constraints I decided this was not high priority and gave my time to more important tasks. Once I learned how to efficiently commit changes to GitLab the commits would take place any time a working change was implemented.
The commits table can be seen below:

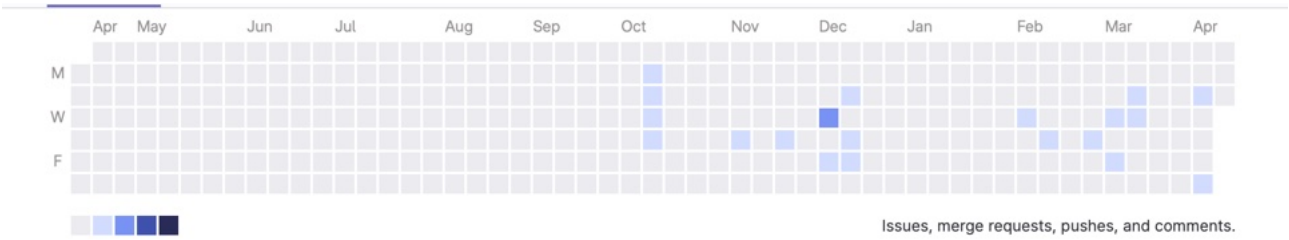


Figure 118: Gitlab Commits

Gantt-Chart

To allow for easy visual progress tracking I have created a Gantt chart using Google Spreadsheets. This Gantt chart would contain all the important Jira Epics, Stories and subtasks. Feedback and Weekly Updates were not included in the Gantt-Chart.
On the Gantt chart you can see which subtasks are linked to which Stories and which Stories are linked to which Epics. I would change the progress % of every issue regularly and implement updates whenever necessary. The Gantt-Chart can be seen on the next page:

CE301 GANTT CHART

PROJECT TITLE	Developing A mobile app using machine learning for cat breed recognition	AUTHOR	Olaf Bogus
START DATE	10/10/2022	OTHER	N/A

Task ID	Linked Issue	TASK TITLE	TASK TYPE	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
C301092-2	N/A	Challenge Week Objectives	Epic	Week 1	Week 1	5 days	100%
C301092-3	C301092-2	Create a Risk Assesement	Story	Week 1	Week 1	5 days	100%
C301092-4	C301092-2	Background Reading Summary	Story	Week 1	Week 1	5 days	100%
C301092-5	C301092-2	Technical Deliverables	Story	Week 1	Week 1	5 days	100%
C301092-8	C301092-2	Project Plan	Story	Week 1	Week 1	5 days	100%
C301092-9	C301092-2	Summer Preparation Record	Story	Week 1	Week 1	5 days	100%
C301092-41	C301092-2	Prepare a Presentation	Story	Week 1	Week 1	5 days	100%
C301092-42	C301092-9	GUI Sketches	Sub-Task	Week 1	Week 1	5 days	100%
C301092-43	C301092-9	Background Reading	Sub-Task	Week 1	Week 1	5 days	100%
C301092-6	C301092-5	Algorithm	Sub-Task	Week 1	Week 1	5 days	100%
C301092-7	C301092-5	Create a GUI	Sub-Task	Week 1	Week 1	5 days	100%
C301092-13	C301092-11	Basic Cat Detection	Sub-Task	Week 1	Week 1	5 days	100%
C301092-18	C301092-5	Simple GUI	Sub-Task	Week 1	Week 1	5 days	100%
C301092-22	C301092-2	Functionality Implementation: Image Processing	Sub-Task	Week 1	Week 1	5 days	100%
C301092-10	N/A	Road to Week 11	Epic	Week 2	Week 11	9 Weeks	100%
C301092-11	C301092-10	Machine Learning	Story	Week 2	Week 11	9 Weeks	100%
C301092-12	C301092-10	App Development	Story	Week 2	Week 11	9 Weeks	100%
C301092-14	C301092-11	Advanced Cat Detection	Sub-Task	Week 4	Week 4	1 Week	100%
C301092-19	C301092-12	GUI Improvements Week 4	Sub-Task	Week 4	Week 4	1 Week	100%
C301092-15	C301092-11	Simple Cat Breed Classification	Sub-Task	Week 5	Week 5	1 Week	100%
C301092-23	C301092-12	Functionality Implementation: Breed Search Screen	Sub-Task	Week 6	Week 6	1 Week	100%
C301092-20	C301092-12	GUI Improvements Week 6	Sub-Task	Week 6	Week 6	1 Week	100%
C301092-16	C301092-11	Interim Cat Breed Classification	Sub-Task	Week 7	Week 7	1 Week	100%
C301092-24	C301092-12	Functionality Implementation: Scan History Screen	Sub-Task	Week 8	Week 8	1 Week	100%
C301092-52	C301092-11	TensorFlow to CoreML	Sub-Task	Week 8	Week 8	1 Week	100%
C301092-17	C301092-11	Advanced Cat Breed Classification	Sub-Task	Week 9	Week 9	1 Week	100%
C301092-20	C301092-12	GUI Improvements Week 9	Sub-Task	Week 9	Week 9	1 Week	100%
C301092-25	C301092-12	Functionality Implementation: Breed Search	Sub-Task	Week 9	Week 9	1 Week	0%
C301092-26	C301092-12	Functionality Implementation: Scan History	Sub-Task	Week 9	Week 9	1 Week	0%
C301092-21	C301092-12	GUI Improvements Week 10	Sub-Task	Week 10	Week 10	1 Week	100%
C301092-27	C301092-12	Cat Breed Data Gathering & Database Assembly	Sub-Task	Week 10	Week 10	1 Week	0%
C301092-28	N/A	Road to Final Report	Epic	Week 16	Week 30	14 Weeks	75%
C301092-29	C301092-28	App Work	Story	Week 16	Week 27	9 Weeks	99%
C301092-33	C301092-28	Report Work	Story	Week 22	Week 29	7 Weeks	33%
C301092-34	C301092-28	Open Day Preparation	Story	Week 23	Week 30	7 Weeks	100%
C301092-68	C301092-29	Machine Learning: Switch Datasets	Sub-Task	Week 16	Week 16	1 Week	100%
C301092-70	C301092-29	Machine Learning: Expand Dataset	Sub-Task	Week 17	Week 17	1 Week	100%
C301092-64	C301092-29	Functionality Implementation: Cat Encyclopedia	Sub-Task	Week 18	Week 20	3 Week	100%
C301092-63	C301092-29	Functionality Implementation: History Screen	Sub-Task	Week 19	Week 20	2 Weeks	100%
C301092-85	C301092-33	First Report Plan	Sub-Task	Week 20	Week 20	1 Week	100%
C301092-31	C301092-29	Machine Learning Training	Sub-Task	Week 21	Week 24	4 Week	100%
C301092-30	C301092-29	Visual Improvements	Sub-Task	Week 22	Week 22	1 Week	100%
C301092-32	C301092-29	Bug Fixes	Sub-Task	Week 22	Week 22	1 Week	100%
C301092-92	C301092-29	User Testing	Sub-Task	Week 23	Week 23	1 Week	100%
C301092-93	C301092-29	Cat Detection Feasibility	Sub-Task	Week 23	Week 23	1 Week	100%
C301092-35	C301092-33	Improved Report Plan	Sub-Task	Week 24	Week 24	1 Week	100%
C301092-30	C301092-34	Poster Creation	Sub-Task	Week 24	Week 24	1 Week	100%
C301092-90	C301092-34	Abstract Creation	Sub-Task	Week 24	Week 24	1 Week	100%
C301092-96	C301092-29	User Testing Part 2	Sub-Task	Week 24	Week 24	1 Week	100%
C301092-33	C301092-98	Chapter 1: Project Introduction	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-99	Chapter 2: Project background research	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-100	Chapter 3: App Design Description	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-101	Chapter 4: App Implementation Description	Sub-Task	Week 24	Week 30	6 Weeks	20%
C301092-33	C301092-102	Chapter 5: Machine Learning: Detection	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-103	Chapter 6: Machine Learning: Classification	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-104	Chapter 7: Project Management	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-33	C301092-105	Chapter 8: Conclusion & further work	Sub-Task	Week 24	Week 30	6 Weeks	0%
C301092-29	C301092-106	Further Improvements	Sub-Task	Week 25	Week 30	6 Weeks	0%
C301092-39	C301092-33	First draft of the report	Sub-Task	Week 25	Week 25	1 Week	0%
C301092-38	C301092-33	Planning out the report	Sub-Task	Week 25	Week 25	1 Week	0%
C301092-65	C301092-29	App Publication	Sub-Task	Week 27	Week 27	1 Week	0%
C301092-40	C301092-33	Final Version of Report	Sub-Task	Week 29	Week 29	1 Week	0%
C301092-36	C301092-34	Demonstration Planning	Sub-Task	Week 30	Week 30	1 Week	0%
C301092-37	C301092-33	Create a Planning Record	Sub-Task	Week 30	Week 30	1 Week	0%

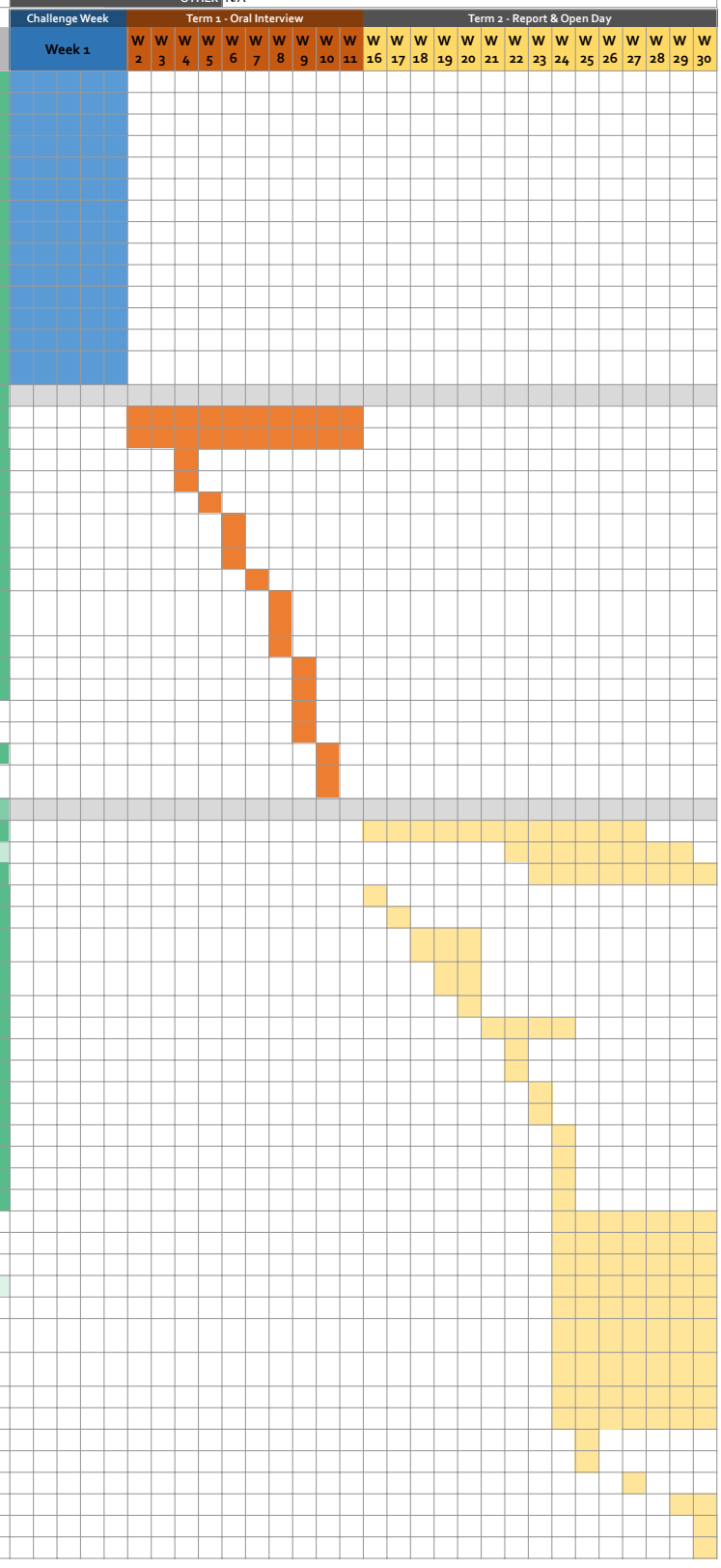


Figure 119: Gantt-Chart(2)

Chapter 7: Conclusion

7.1 Technical Accomplishments

This project was able to accomplish all the planned technical achievements.
This being a fully functioning mobile app operating on iOS and is optimised for iPhones.
The app allows for Cat Breed classification from existing photos as well as from a photo taken within the app. The classification results and the photo is then stored in an accessible way to the user. The app contains an encyclopaedia page that contains key information about each supported cat breed as well as external links for more detailed information. The app contains a history screen which presents all the saved data from previous classification results. The data can be deleted if the user wishes. The photos are compressed before being saved.

The App utilises two machine learning models. One model is used for cat breed classification. It supports 20 breeds and has an accuracy of 94%. The model was trained on a manually expanded dataset.
The other model used by the app performs binary classification to determine wether the supplied photo contains a cat or not.
These models were converted to a coreML model in order for it to work with the iOS app.

7.2 Summary Thoughts

The project aimed to develop an iOS app capable of utilising machine learning models for purposes of identifying cat breeds from photos. The app was to be user friendly and have functionalities such as an encyclopaedia that would present the user with key information pertaining the cat breeds supported by machine learning models, saving data processed by the user and presenting it in an accessible way. The machine learning models were to be of high accuracy and wide classification capabilities.

These aims were all achieved fully. Taking an enormous amount of time, effort and focus.

The project was managed using Jira. It being a key tool that allowed for splitting extremely complex tasks into small single issues of much reduced complexity.

Throughout the duration of the project I have felt extreme pressure to achieve best results possible.

Not having a lot of experience with iOS software development or Machine Learning every task I tackled involved a learning process. I had to identify the task at hand, see examples of similar issues being implemented through code, plan how to implement it in my program and then actually code it. Some tasks would turn out much easier than previously assumed and some were much more complex in their implementation than ever expected.

Having known now what I have learned in the duration of the project I can confidently say I could implement the same quality software within a month.
The skills I have gained thanks to this project are invaluable.
The research and studies were the most time consuming part of the project.

Having achieved all of the planned goals does not mean the project could not be improved.
With more time the application could be immensely improved.
The dataset that the classification model was trained on definitely needs expanding to increase accuracy and supported breeds.
The 2 label model used for recognising if the supplied image contains a cat could be implemented through a different model or trained on a better dataset. During the final weeks of the summer term I contemplated implementing a model capable of detecting multiple cats on the image and then classifying each detected cat separately. This was determined not feasible in the given timeframe.
The app could have expanded functionality such as an inbuilt social network for sharing cat images with other users, the encyclopaedia page could be enriched to include more information for each breed and more images. There are numerous features that could be considered to be added to the app. I would have loved to be able to publish the app on the iOS App Store but was unable to do so due to the costs.

I am very proud of the work I was able to accomplish.

References

[1] W. Rawat and Z. Wang, “Deep convolutional neural networks for Image Classification: A Comprehensive Review,” *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional Neural Networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[3] C. Szegedy, W. Liu, J. Yangqing, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[4] K. Simonyan and A. Zisserman, “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION,” *ICLR 2015*, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” *International Conference on Machine Learning*, 2019.

[7] R. Peden, “Cat or not - an image classifier using Python and keras,” *CodeProject*, 01-May-2019. [Online]. Available: https://www.codeproject.com/Articles/4023566/Cat-or-Not-An-Image-Classifier-using-Python-and-Ke.

[8] S. Gulati, “Creating and deploying a cat-dog image classifier using tensorflow and streamlit-part 1,” *Medium*, 15-Jul-2021. [Online]. Available: https://medium.com/analytics-vidhya/creating-and-deploying-a-cat-dog-image-classifier-using-tensorflow-and-streamlit-part-1-2672fc184601.

[9] V. Valkov, “Building a cat detector using convolutional neural networks-tensorflow for hackers (part III),” *Medium*, 13-Jun-2017. [Online]. Available: https://venelinvalkov.medium.com/tensorflow-for-hackers-part-iii-convolutional-neural-networks-c077618e590b.

[10] P. Kalkman, “Recognizing cats and dogs with tensorflow,” *Medium*, 11-Jan-2021. [Online]. Available: https://towardsdatascience.com/recognizing-cats-and-dogs-with-tensorflow-105eb56da35f.

[11] D. F. Team, “Cats vs dogs classification (with 98.7% accuracy) using CNN Keras - Deep Learning Project for Beginners,” *DataFlair*, 25-Aug-2021. [Online]. Available: https://data-flair.training/blogs/cats-dogs-classification-deep-learning-project-beginners/.

[12] J. Brownlee, “How to classify photos of dogs and cats (with 97% accuracy),” *MachineLearningMastery.com*, 07-Dec-2021. [Online]. Available: https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos-of-dogs-and-cats/.

[13] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, “The Oxford-IIIT Pet Dataset,” *Visual Geometry Group - University of Oxford*. [Online]. Available: https://www.robots.ox.ac.uk/~vgg/data/pets/.

[14] ma7555, “Cat breeds dataset,” *Kaggle*, 12-Dec-2019. [Online]. Available: https://www.kaggle.com/datasets/ma7555/cat-breeds-dataset.

[15] N. Mohtar, “Cat Breed Classification system using CNN - nurafiqah mohtar,” *YouTube*, 01-Feb-2021. [Online]. Available: https://www.youtube.com/watch?v=rNgrYfaiYCs&t=181s.

[16] N. Thammarat, “Inceptionv3 with ROC- AUC 92%,” *Kaggle*, 07-Mar-2023. [Online]. Available: https://www.kaggle.com/code/thammaratnunum/inceptionv3-with-roc-auc-92-e459a2.

[17] N. Renotte, “Build a deep CNN image classifier with any images,” *YouTube*, 25-Apr-2022. [Online]. Available: https://www.youtube.com/watch?v=jztwpsIzEGc.

[18] I. Developer, “Dog breed classification using Tensorflow Keras | Building an image classifier,” *YouTube*, 07-Mar-2020. [Online]. Available: https://www.youtube.com/watch?v=Ku6wkRQIYuc.

[19] M. Ganeson, “Dog breed classification using CNNS,” *Medium*, 13-Aug-2021. [Online]. Available: https://mitraganeson.medium.com/dog-breed-classification-using-cnns-73596aa37a06.

[20] “Dog breed identification,” *Kaggle*. [Online]. Available: https://www.kaggle.com/competitions/dog-breed-identification/code.

[21] “Overfit and underfit : Tensorflow Core,” *TensorFlow*. [Online]. Available: https://www.tensorflow.org/tutorials/keras/overfit_and_underfit.

[22] N. Schlüter, “Don't Overfit!-how to prevent overfitting in your deep learning models,” *Medium*, 05-Jun-2019. [Online]. Available: https://towardsdatascience.com/dont-overfit-how-to-prevent-overfitting-in-your-deep-learning-models-63274e552323.

[23] U. Vijay, “Early stopping to avoid overfitting in neural network- Keras,” *Medium*, 13-Oct-2020. [Online]. Available: https://medium.com/zero-equals-false/early-stopping-to-avoid-overfitting-in-neural-network-keras-b68c96ed05d9#:~:text=Too%20many%20epochs%20can%20lead,improving%20on%20the%20validation%20dataset.

[24] MammothInteractive, “Train a machine learning model with createml | super simple,” *YouTube*, 02-Nov-2021. [Online]. Available: https://www.youtube.com/watch?v=nCwIsGjN9XQ.

[25] Identify, “Let's create a simple login screen for iPhone in Xcode (swiftui),” *YouTube*, 16-Nov-2021. [Online]. Available: https://www.youtube.com/watch?v=l7obVQObdRM.

[26] iOS Academy, “Swiftui 4: New Photo Picker tutorial – WWDC 2022,” *YouTube*, 11-Jul-2022. [Online]. Available: https://www.youtube.com/watch?v=crULPMS7Uxs.

[27] Cracking Code with Dave, “Build and run app from Xcode onto actual device in Xcode 12.15.1,” *YouTube*, 21-Sep-2021. [Online]. Available: https://www.youtube.com/watch?v=TlgumE2xe_E.

[28] Indently, “Splashscreen for IOS in Swiftui tutorial 2022 (xcode),” *YouTube*, 25-Jan-2022. [Online]. Available: https://www.youtube.com/watch?v=0ytO3wCRKZU.

[29] Intently, “How to create a bottom navigation bar with TabView in Xcode (swiftui / ios),” *YouTube*, 22-Oct-2021. [Online]. Available: https://www.youtube.com/watch?v=TgvYFfCjDMo.

[30] iOS Academy, “Swift for beginners: Select photo from library IOS (2020),” *YouTube*, 30-Jan-2020. [Online]. Available: https://www.youtube.com/watch?v=yggOGEzueFk.

[31] S. Allen, “SWIFTUI photo picker - compressed images, uiviewcontrollerrepresentable,” *YouTube*, 29-Jul-2021. [Online]. Available: https://www.youtube.com/watch?v=V-kSSjh1T74.

[32] iOS Academy, “How to add images to IOS app - swift (2020),” *YouTube*, 02-Feb-2020. [Online]. Available: https://www.youtube.com/watch?v=Tb9J08y5a4w.

[33] F. Hussain, “Tutorial 31: How to upload and display image on IOS app | how to add images to IOS app swift 2021,” *YouTube*, 14-Jun-2021. [Online]. Available: https://www.youtube.com/watch?v=jaagKolaDSA.

[34] iOS Academy, “IOS 15 photo picker tutorial (2021, Xcode 13, IOS 15) - IOS for Beginners,” *YouTube*, 16-Jun-2021. [Online]. Available: https://www.youtube.com/watch?v=LlZUQW3Zj9c.

[35] P. Hudson, “Importing an image into swiftui using phpickerviewcontroller – instafilter swiftui tutorial 9/12,” *YouTube*, 03-Dec-2021. [Online]. Available: https://www.youtube.com/watch?v=-4wBQSr-3yo.

[36] P. Hudson, “Wrapping a uiviewcontroller in a swiftui view – Instafilter Swiftui tutorial 5/12,” *YouTube*, 01-Dec-2021. [Online]. Available: https://www.youtube.com/watch?v=qKAlsts4qFA.

[37] MammothInteractive, “Load coreml model into Xcode | Super Simple,” *YouTube*, 24-Oct-2021. [Online]. Available: https://www.youtube.com/watch?v=uRFxyk-xGE4.

[38] MammothInteractive, “Use coreml model in swiftui,” *YouTube*, 25-Oct-2021. [Online]. Available: https://www.youtube.com/watch?v=8Gl33PpOlJg.

[39] MammothInteractive, “Compile coreml model in Xcode,” *YouTube*, 27-Oct-2021. [Online]. Available: https://www.youtube.com/watch?v=qIX5vHhxjR8.

[40] MammothInteractive, “Test createml model with coreml in an app | super simple,” *YouTube*, 03-Nov-2021. [Online]. Available: https://www.youtube.com/watch?v=UaZfa_rddME.

[41] iOS Academy, “CoreML basics in IOS (swift 5, Machine Learning, Xcode 12) - 2022 IOS development,” *YouTube*, 20-Mar-2021. [Online]. Available: https://www.youtube.com/watch?v=OxKHt1NwOHw.

[42] S. Lynch, “My images 1: Photo Picker and camera in Swiftui,” *YouTube*, 26-Sep-2021. [Online]. Available: https://www.youtube.com/watch?v=yMC16EZHwZU.

[43] Kavsoft, “Swiftui popup image picker - custom popup's, image picker modifier - xcode 13 - swiftui tutorials,” *YouTube*, 24-May-2022. [Online]. Available: https://www.youtube.com/watch?v=dQUgCyb-OMU.

[44] Azamsharp, “The Complete Guide for Integrating Camera and Photo Library in Swiftui,” *YouTube*, 11-Feb-2020. [Online]. Available: https://www.youtube.com/watch?v=Y-65T0YBOm4.

[45] CodeWithChris, “How to create an app Icon (2019),” *YouTube*, 28-Aug-2018. [Online]. Available: https://www.youtube.com/watch?v=bOjoBgtjwt4.

[46] iOS Academy, “How to create launch screen / image in swift 5 and Xcode 11,” *YouTube*, 18-Apr-2020. [Online]. Available: https://www.youtube.com/watch?v=FdiqRol8weU.

[47] Azamsharp, “Saving images using Core Data,” *YouTube*, 21-Dec-2021. [Online]. Available: https://www.youtube.com/watch?v=XqqIwJAqNZs.

[48] “Creating a core data model,” *Apple Developer Documentation*. [Online]. Available: https://developer.apple.com/documentation/coredata/creating_a_core_data_model.

[49] B. Jacobs, “Fetching records with core data: Type methods,” *Cocoacasts*. [Online]. Available: https://cocoacasts.com/fetching-records-with-core-data-type-methods-in-swift.

[50] “*Core Data Programming Guide: Fetching objects”*, 27-Mar-2017. [Online]. Available: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CoreData/FetchingObjects.html.

[51] P. Hudson, “How to combine core data and swiftui,” *Hacking with Swift*, 12-May-2022. [Online]. Available: https://www.hackingwithswift.com/books/ios-swiftui/how-to-combine-core-data-and-swiftui.

[52] iOS Academy, “Swiftui dynamic list app (Xcode 12, 2021, swiftui 2.0) - IOS development,” *YouTube*, 13-Feb-2021. [Online]. Available: https://www.youtube.com/watch?v=JCez4mclzdQ.

[53] “Xcode 11.4.1 - loading a launchscreen image from assets folder,” *Stack Overflow*, 01-Feb-1967. [Online]. Available: https://stackoverflow.com/questions/61662245/xcode-11-4-1-loading-a-launchscreen-image-from-assets-folder.

[54] P. Hudson, “How to delete core data objects from SWIFTUI views,” *Hacking with Swift*, 19-Jan-2023. [Online]. Available: https://www.hackingwithswift.com/quick-start/swiftui/how-to-delete-core-data-objects-from-swiftui-views.

[55] P. Hudson, “How to delete a core data object,” *Hacking with Swift*, 28-May-2019. [Online]. Available: https://www.hackingwithswift.com/read/38/9/how-to-delete-a-core-data-object.

[56] “Swift 3 core data delete object,” *Stack Overflow*, 01-Mar-1963. [Online]. Available: https://stackoverflow.com/questions/38017449/swift-3-core-data-delete-object.

[57] P. Hudson, “Deleting from a core data fetch request,” *Hacking with Swift*, 24-Nov-2021. [Online]. Available: https://www.hackingwithswift.com/books/ios-swiftui/deleting-from-a-core-data-fetch-request.

[58] “How to delete core data SWIFTUI?,” *Stack Overflow*, 01-Apr-1968. [Online]. Available: https://stackoverflow.com/questions/68432307/how-to-delete-core-data-swiftui.

[59] “Delete/reset all entries in core data?,” *Stack Overflow*, 01-Apr-1956. [Online]. Available: https://stackoverflow.com/questions/1077810/delete-reset-all-entries-in-core-data.

[60] “SWIFTUI update of deleted CoreData entities from different Tab,” *SwiftUI update of deleted CoreData... | Apple Developer Forums*. [Online]. Available: https://developer.apple.com/forums/thread/670564.

[61] J. Codeos, “How to save, fetch, update and delete data from core data using swift: John Codeos - blog with free IOS & Android Development Tutorials,” *John Codeos*, 13-Jun-2022. [Online]. Available: https://johncodeos.com/how-to-use-core-data-in-ios-using-swift/.

[62] Azamsharp, “Core data with swiftui (create, read, update and delete),” *YouTube*, 14-Feb-2021. [Online]. Available: https://www.youtube.com/watch?v=_ui7pxU1rNI.

[63] “Core data: Quickest way to delete all instances of an entity,” *Stack Overflow*, 01-Jun-1956. [Online]. Available: https://stackoverflow.com/questions/1383598/core-data-quickest-way-to-delete-all-instances-of-an-entity.

[64] “Core Data Object not nil after deleting from context,” *Stack Overflow*, 01-Jan-1968. [Online]. Available: https://stackoverflow.com/questions/67008050/core-data-object-not-nil-after-deleting-from-context.

[65] P. Hudson, “How to delete core data objects from SWIFTUI views,” *Hacking with Swift*, 19-Jan-2023. [Online]. Available: https://www.hackingwithswift.com/quick-start/swiftui/how-to-delete-core-data-objects-from-swiftui-views.

[66] “How to get the size of data present in coredata store?,” *Stack Overflow*, 01-Mar-1961. [Online]. Available: https://stackoverflow.com/questions/24178776/how-to-get-the-size-of-data-present-in-coredata-store.

[67] “Compress a UIImage,” *SwiftUI Advanced Handbook - Design+Code*. [Online]. Available: https://designcode.io/swiftui-advanced-handbook-compress-a-uiimage/.

[68] “In swift, how to reduce an image file size to a specific size? e.g 1MB,” *In Swift, how to reduce an image f... | Apple Developer Forums*. [Online]. Available: https://developer.apple.com/forums/thread/67564.

[69] “Date() formatting in SWIFTUI ▸ swiftlyrush,” *SwiftlyRush*, 29-Mar-2022. [Online]. Available: https://www.swiftlyrush.com/date-formatting-in-swiftui/.

[70] B. Pasquier, “How to display date and time in Swiftui,” *Benoit Pasquier*, 17-Jan-2021. [Online]. Available: https://benoitpasquier.com/date-formatter-swiftui/.

[71] S. Wongpatcharapakorn, “How to change background color of button in Swiftui,” *Sarunw*, 29-Dec-2022. [Online]. Available: https://sarunw.com/posts/swiftui-button-background-color/.

[72] “Swiftui label tutorial – how to create and use label in swiftui,” *Simple Swift Guide*, 30-Dec-2020. [Online]. Available: https://www.simpleswiftguide.com/swiftui-label-tutorial-how-to-create-and-use-label-in-swiftui/.

[73] S. Simonng, “Swiftui Tabview Introduction and tab bar customization,” *AppCoda*, 03-Oct-2020. [Online]. Available: https://www.appcoda.com/swiftui-tabview/.

[74] “Links in Swiftui,” *Swift Anytime*. [Online]. Available: https://www.swiftanytime.com/blog/links-in-swiftui.

[75] A. Pereira, “Swiftui Button Tutorial: Customization,” *kodeco.com*. [Online]. Available: https://www.kodeco.com/34851726-swiftui-button-tutorial-customization.

[76] F. Němeček, “How to get file size using FileManager + formatting,” *Filip Němeček*. [Online]. Available: https://nemecek.be/blog/22/how-to-get-file-size-using-filemanager-formatting.

[77] P. Hudson, “How to create a Tappable button,” *Hacking with Swift*, 02-Dec-2022. [Online]. Available: https://www.hackingwithswift.com/quick-start/swiftui/how-to-create-a-tappable-button.

[78] P. Hudson, “How to run some code when state changes using onChange(),” *Hacking with Swift*, 01-Dec-2022. [Online]. Available: https://www.hackingwithswift.com/quick-start/swiftui/how-to-run-some-code-when-state-changes-using-onchange.

[79]“How does the Imagepicker get analyzed by the MLMODEL?,” *Stack Overflow*, 01-May-1967. [Online]. Available: https://stackoverflow.com/questions/63458277/how-does-the-imagepicker-get-analyzed-by-the-mlmodel.

[80] “Advanced guide to inception V3 | cloud TPU | google cloud,” *Google*. [Online]. Available: https://cloud.google.com/tpu/docs/inception-v3-advanced.

[81] Larxel, “Dog and cat detection,” *Kaggle*, 12-May-2020. [Online]. Available: https://www.kaggle.com/datasets/andrewmvd/dog-and-cat-detection.

[82] “Cat breeds,” *Cat Breeds - Cat Breed Library | Purina UK*. [Online]. Available: https://www.purina.co.uk/find-a-pet/cat-breeds?page=%2C1.

[83] “Sand Cat,” *Smithsonian's National Zoo*, 24-May-2022. [Online]. Available: https://nationalzoo.si.edu/animals/sand-cat.

Appendixes

Abstract

Title: Orion: Cat Scanner and Encyclopaedia - an iOS App Implementing Machine Learning for Cat Breed Classification
Name: Olaf Bogus
Reg No: 2006230

Supervisor: John Gan Category: AI/Machine Learning

Machine Learning allows us to achieve things never possible before.
I have implemented machine learning in a mobile app that helps cat lovers learn more about their feline best friends.

Orion: Cat Scanner & Encyclopaedia analyses a photo supplied from either your camera or library. It checks if the photo has a cat in it and matches cat breeds.

This is done using a Keras model with InceptionV3 as the convolutional base layer.

The classification model can detect up to 20 cat breeds with following performance scores: Test accuracy is 90.40 %
F1 score is 89%
ROC- AUC score is 0.99

It has been trained on a custom assembled dataset consisting of 200 photos per cat breed. The App has been developed with Xcode using SwiftUI.

The Orion App will tell you the top 3 matched breeds as well as details for every breed with its cat breed encyclopaedia that contains photos, key information and external links.
It saves every analyzed photo and the assigned matches in the app history which can be accessed and edited at any time.

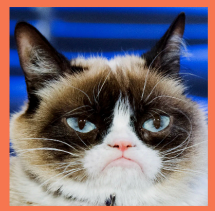
University of Essex

Orion: Cat Scanner & Encyclopedia - an iOS App Implementing Machine Learning for Cat Breed Classification

School of Computer Science and Electronic Engineering

Olaf Bogus supervised by Professor John Gan

Learn More About Your Cat!

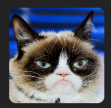


This cat shares the features of:

Ragdoll : 50.6%
Birman : 44.3%
Siamese : 3.3%

Look Through Your History!

CAT SCANNER HISTORY



Mar 13, 2023
3:00:17 PM

Qrion

Cat Scanner and Encyclopedia

Scan Up To 20 Different Cat Breeds!

Created with Xcode using SwiftUI
Implemented with a custom Keras Tensorflow model based on InceptionV3
Achieving:
Test accuracy of 90.40%
F1 Score of 89%
ROC-AUC Score of 0.99

Learn More About Their Needs & Behavior With The Encyclopedia Page!

Persian

They might have a grumpy looking face, but make no mistake, the Persian cat is as sweet and easy going as it can get. The fancy felines are loving and charming creatures that would get along with the entire family if introduced to them as kittens. But beauty is not always effortless, so Persian cats do require extra attention – find out more about them.

Source: https://www.purina.co.uk/find-a-pet/cat-breeds/persian-long-hair

Poster