IdentiFlora

Houseplant Identification and Care Application

Shape

Description automatically generated with medium confidence

Joseph Jarvis - 18018718

UXCFXK-30-3

Digital Systems Project 2021/2022

Text

Description automatically generated

# Abstraction

Despite the extensive research done into wild and agricultural plant identification using artificial intelligence, very little research has been done into the identification and classification of houseplants.

The goal of this research and this project is to recodify these issues, with the creation of houseplant identification and care application, that using an AI will identify a given plant, providing information on how to best care for the identified plant. This project will cover the deduction of the best AI model for houseplant identification, how to best optimise an AI model for use on mobile and IoT devices, the building and training of the deduced AI mode, the building of an application front end as well as the creation of a database of plant care information.

This project concluded with the implementation of the IdentiFlora Houseplant identification and care Android application, consisting of a CNN model optimised using Adamax optimisation and a SQLite database, which could effective identify eleven different species of houseplant with an average training accuracy of 98.56%, validation accuracy of 96.36%, and a testing accuracy of 92.73%.

# Acknowledgments

I would like to thank Shelan Jeawak for their support and guidance throughout the project.

I would also like to thank Martin Serpell for their guidance with the project and for imparting their experience on how to best undertake an academic research project such as this.

I would finally like to thank Charles, Jacob, and Benedict for putting up with my houseplant obsession and their infinite patience with my slow descent into madness due to AI.

Table Of Contents

[Abstraction 2](#_Toc100842986)

[Acknowledgments 3](#_Toc100842987)

[Table Of Figures 7](#_Toc100842988)

[Table of Tables 8](#_Toc100842989)

[List of Abbreviations 8](#_Toc100842990)

[1 Introduction 9](#_Toc100842991)

[1.1 Project Summary 9](#_Toc100842992)

[1.2 The Real-World Problem 9](#_Toc100842993)

[1.3 Aims & Objectives 9](#_Toc100842994)

[2 Research and Literature Review 10](#_Toc100842995)

[2.1 Plant Identification 10](#_Toc100842996)

[2.2 Using AI to Identify Plants 10](#_Toc100842997)

[2.2.1 Evaluation of ANNs 10](#_Toc100842998)

[2.2.2 Evaluation of CNNs 11](#_Toc100842999)

[2.2.3 Evaluation of SVMs 12](#_Toc100843000)

[2.2.4 Evaluation of AI Methods Conclusion 12](#_Toc100843001)

[2.3 Optimisation Methods for a CNN 13](#_Toc100843002)

[2.4 Integrating Neural Networks with IoT/Mobile Devices 13](#_Toc100843003)

[2.5 Ethical Considerations 13](#_Toc100843004)

[3 Requirement Analysis 14](#_Toc100843005)

[3.1 Introduction 14](#_Toc100843006)

[3.1.1 Priority System 14](#_Toc100843007)

[3.1.2 Justification of Requirements 14](#_Toc100843008)

[3.2 User Stories 14](#_Toc100843009)

[3.3 Functional Requirements 15](#_Toc100843010)

[3.3.1 Functional Requirements of the Neural Network, User Interface, and Database 15](#_Toc100843011)

[3.4 Non-Functional Requirements 16](#_Toc100843012)

[3.5 Acceptance testing 17](#_Toc100843013)

[4 Methodology 18](#_Toc100843014)

[4.1 Chosen Methodology 18](#_Toc100843015)

[4.2 Project Plan and Timeline 18](#_Toc100843016)

[4.2.1 Gantt chart 18](#_Toc100843017)

[4.2.2 Hierarchical Breakdown 19](#_Toc100843018)

[5 Design 20](#_Toc100843019)

[5.1 System Architecture 20](#_Toc100843020)

[5.2 Design of the Neural Network - Prototype CNN 21](#_Toc100843021)

[5.3 Design of the Mobile Interface 22](#_Toc100843022)

[5.3.1 Wireframes 22](#_Toc100843023)

[5.3.2 Colour Schemes 23](#_Toc100843024)

[5.3.3 Composites 25](#_Toc100843025)

[5.3.4 Assets 26](#_Toc100843026)

[5.4 Design of the Database 27](#_Toc100843027)

[6. Implementation & Testing 28](#_Toc100843028)

[6.1 CNN Implementation 28](#_Toc100843029)

[6.1.1 Overview 28](#_Toc100843030)

[6.1.2 Datasets 28](#_Toc100843031)

[6.1.3 Image Pre-Processing 28](#_Toc100843032)

[6.1.4 Optimisation Algorithm Comparison 29](#_Toc100843033)

[6.1.5 Initial CNN Training 30](#_Toc100843034)

[6.1.6 Final Product 31](#_Toc100843035)

[6.1.7 Problems and Challenges with the Development of the CNN 31](#_Toc100843036)

[6.2 Mobile Application Implementation 32](#_Toc100843037)

[6.2.1 Overview 32](#_Toc100843038)

[6.2.2 Creating the Mobile Application 32](#_Toc100843039)

[6.2.3 Camera Utilisation and Integration 33](#_Toc100843040)

[6.2.4 Final Product 35](#_Toc100843041)

[6.2.5 Problems and Challenges with the Development of the Mobile Interface 36](#_Toc100843042)

[6.3 Database Implementation 38](#_Toc100843043)

[6.3.1 Overview 38](#_Toc100843044)

[6.3.2 Creating the SQLite Database 38](#_Toc100843045)

[6.3.3 Implementing Plant Data into the Database 38](#_Toc100843046)

[6.3.4 Final Product 39](#_Toc100843047)

[6.3.5 Problems and Challenges with Building the Database 39](#_Toc100843048)

[6.4 Component Integration 40](#_Toc100843049)

[6.4.1 Overview 40](#_Toc100843050)

[6.4.2 Integrating the CNN into the Mobile Application 40](#_Toc100843051)

[6.4.3 Integrating the Database into the Mobile Application 41](#_Toc100843052)

[6.4.4 Final Product 42](#_Toc100843053)

[6.3.4 Problems and Challenges with Component Integration 43](#_Toc100843054)

[6.5 System Testing 45](#_Toc100843055)

[6.5.1 Acceptance Testing 45](#_Toc100843056)

[6.5.2 AI Performance Testing on Real-World Data 45](#_Toc100843057)

[7. Project Evaluation 46](#_Toc100843058)

[7.1 Evaluation of the Research Phase 46](#_Toc100843059)

[7.1.1 What Went Well? 46](#_Toc100843060)

[7.1.2 Limitations 46](#_Toc100843061)

[7.1.3 Improvements and Lessons Learnt 46](#_Toc100843062)

[7.2 Evaluation of the Design Phase 47](#_Toc100843063)

[7.2.1 What Went Well? 47](#_Toc100843064)

[7.2.2 Limitations 47](#_Toc100843065)

[7.2.3 Improvements and Lessons Learnt 47](#_Toc100843066)

[7.3 Evaluation of the Implementation Phase 48](#_Toc100843067)

[7.3.1 What Went Well? 48](#_Toc100843068)

[7.3.2 Limitations 48](#_Toc100843069)

[7.3.3 Improvements and Lessons Learnt 48](#_Toc100843070)

[7.4 Evaluation of the Testing Phase 49](#_Toc100843071)

[7.4.1 What Went Well? 49](#_Toc100843072)

[7.4.2 Limitations 49](#_Toc100843073)

[7.4.3 Improvements and Lessons Learnt 49](#_Toc100843074)

[8. Conclusion and Further Work 50](#_Toc100843075)

[8.1 Conclusion 50](#_Toc100843076)

[8.2 Further Work 50](#_Toc100843077)

[8.2.1 Expansion of the Database, the AI Model, & the Datasets 50](#_Toc100843078)

[8.2.2 Mobile Application Quality of Life Improvements 50](#_Toc100843079)

[8.2.3 System Implementation on Other Platforms 50](#_Toc100843080)

[9. References / Bibliography 52](#_Toc100843081)

[10 Appendix 55](#_Toc100843082)

[10.1 Requirement Analysis 55](#_Toc100843083)

[10.1.1 Additional User stories 55](#_Toc100843084)

[10.1.2 Additional Functional Requirements 56](#_Toc100843085)

[10.1.3 Addition Non-Functional Requirements 57](#_Toc100843086)

# Table Of Figures

Figure 1: Project Gantt Chart 18

Figure 2: Project Hierarchical Breakdown 19

Figure 3 : IdentiFlora System Architecture 20

Figure 4 : Initial Protoype CNN structure 21

Figure 5: Initial wireframe mockups 22

Figure 6: Final version of the mobile interface wireframe 23

Figure 7 : Colour scheme 1 23

Figure 8 : Colour scheme 2 24

Figure 9 : Mobile Interface Composites 25

Figure 10 : Initial digital mock-ups of potential logos 26

Figure 11 : Logo Design 1 26

Figure 12 : Logo Design 2 27

Figure 13: The Plant Care inforatmion Database Structure Diagram 27

Figure 14: Prototype CNN TensorFlow Model for optimisation algorithm testing 29

Figure 15 : Model compiler using Adam optimisation and results 30

Figure 16: Model compiler using Adamax optimisation and results 30

Figure 17: Results of training the first TensorFlow Lite implementation of the CNN 31

Figure 18: Results of training the second TensorFlow Lite implementation of the CNN plant classification AI 31

Figure 19 : IdentiFlora App Icon 32

Figure 20: Application interface as viewed in Android Studio .xml design editor 33

Figure 21: Code used to open the default Android camera application 34

Figure 22: Camera open through the application 34

Figure 23: Screenshot of the mobile application at the end of the mobile application development phase 35

Figure 24: Screenshot of an image taken with the camera not displaying correctly 36

Figure 25: Screenshot of the outcome of the initial fix 36

Figure 26: Code used to rotate the image to match the rotation of the device using EXIF image tag data 37

Figure 27: Database creation code 38

Figure 28: Plant information and care database visualised using SQLite Viewer 39

Figure 29: Code used to obtain the probabilistic result of which plant is present in the provided image 40

Figure 30: Code for displaying content in the table view from the database based on CNN output 41

Figure 31: Screenshot of the final application, shown here identifying a nerve plant 42

Figure 32: Empty Database error message 43

Figure 33: Database made in python before porting over to the Android application 43

Figure 34: The exported database that was being opened by the Android application 43

Figure 35: Code used to copy the database from the assets folder so it could then be used 44

# Table of Tables

[Table 1: User Stories 14](#_Toc100840423)

[Table 2: A Sample of the Functional Requirements 15](#_Toc100840424)

[Table 3: A Sample of the Non-Functional Requirements 16](#_Toc100840425)

[Table 4: Acceptance Tests 17](#_Toc100840426)

[Table 5: Testing the Acceptance Tests 45](#_Toc100840427)

[Table 6: Testing the Real-World Performance of the CNN 45](#_Toc100840428)

[Table 7: The Additional User Stories 56](#_Toc100840429)

[Table 8: The Additional Functional Requirements 57](#_Toc100840430)

[Table 9: The Additional Non-Functional Requirements 58](#_Toc100840431)

# List of Abbreviations

|  |  |
| --- | --- |
| Abbreviation | Description |
| CNN | Convolutional neural network |
| ANN | Artificial neural network |
| SVM | Support vector machine |
| AI | Artificial Intelligence |
| DNN | Deep neural network |
| EXIF | Exchangeable Image File Format |
| iOS | iPhone Operating System |
| IoT | Internet of Things |

# 1 Introduction

## 1.1 Project Summary

The purpose of this project is to build an AI system that can identify a large variety of common houseplants. This system will provide information on how to care for the identified plant. This system must be lightweight and easily accessible to the average user, so must take the form of a mobile application, that when provided an image of a houseplant will attempt to identify it, returning what the AI determined the houseplant to be, along with any corresponding information that is relevant to the effective care of that identified houseplant.

## 1.2 The Real-World Problem

Humans have been using indoor plants for decoration for a large proportion of our history, as stated by Bringslimark, Hartig, and Patil, “Written evidence shows that the Egyptians brought plants indoors in the 3rd century BC, and the ruins of Pompeii revealed that interior plants were used there more than 2000 years ago” (Bringslimark, Hartig, and Patil, 2009). With this being the case combined with the information that there are a large variety of different plant species with “350,000 accepted species, of which 325,000 are flowering plants” (Royal Botanic Gardens, 2020) and the ever-increasing demand for houseplants, ensuring the proper care is vital of these plants to allow them to thrive. To ensure their proper care these plants must first be identified, which will be difficult to achieve for non-experts in the field, so a way of simulating expert knowledge in an easily accessible format is needed.

## 1.3 Aims & Objectives

* To create an artificial intelligence system that can identify a large variety of different houseplants
* Create a database of plant species containing plant care information
* Create a mobile user interface that will allow the user to upload an image of their plant for the AI system to identify, displaying relevant care information for the identified species of plant
* Identify the most effective artificial intelligence method for plant identification
* Identify the most effective ways to optimise an artificial intelligence for use on a mobile device
* Identify how to effectively integrate an artificial intelligence into a mobile/IoT platform

# 2 Research and Literature Review

## 2.1 Plant Identification

As concluded by Cham et al. (2015) plants can be identified through their flowers, fruits, stem, bark, and leaves, with leaves being the most effective as a leaf’s features tend to be universal and persistent over each example of a given plant species. The primary features of the leaf that are useful for identification consist of leaf shape and how the leaves are structured together on the plant, known as the plant's canopy structure, as stated by Jones et al. the “canopy structure and leaf shape have been key features for plant species identification” (Jones et al., 2006). Furthermore, as concluded by Awang et al. (2013) along with shape, the colour and texture features of a plant's leaf were also effective indicators of what species the plant was.

## 2.2 Using AI to Identify Plants

As concluded above, any AI system implemented must effectively identify a plant based on leaves and overall canopy structure meaning Image recognition is required. Image recognition and classification with AI can be done in a few ways, as concluded by Al-Murad, Islam, and Raj (2017) in their analysis of current AI methods for image recognition and classification. One method is to use an ANN, another would be to use a CNN. Finally, they concluded that a SVM is also a suitable AI method.

To move forward, each method must be evaluated to conclude which would be most suitable for plant recognition through the evaluation of previous work done.

### 2.2.1 Evaluation of ANNs

As concluded by Aakif and Khan (2015) ANNs can be used to great effect, with them achieving results of over 96% accuracy with their implementation of an ANN using backpropagation, stating that “we have tested it on three different sets and achieved accuracy greater than 96%.” (Aakif and Khan, 2015). Furthermore, Macario, Oliveira, and Pacifico (2018) also achieved a similar result using a multi-layered perceptron, a type of feedforward ANN, also using back propagation, with their implementation achieving a similar accuracy in real-world tests, “the algorithm was able to achieve an average accuracy of 97.16%”( Macario, Oliveira, and Pacifico, 2018).

However as stated by Choo, Huang, Liu, and Wang (2017), as well as Lu and Wang (2005) ANNs, have two main disadvantages, the first drawback being that “ANN-based classifications are slow as these are black-box models with a gradient descend optimization and too many parameters.” (Choo et al, 2017) and the second being that ANNs have a prevalent issue with overfitting, more so on average than other methods, this results in ANNs if not carefully implemented becoming less effective when handling real-world data outside their initial training and validation datasets.

### 2.2.2 Evaluation of CNNs

As concluded by Arfin, Hossain, Islam, and Rabby (2019) CNNs can be used to great effect, with their implementation of a CNN with the addition of ADAM optimization, which, as concluded by Zhang (2018), is an adaptive optimisation algorithm which adaptively adjusts the learning rate of a deep neural network (like a CNN) to determine and set the most optimal learning rate for each parameter of the DNN, with them stating that “The model ran for 50 epoch resulted training accuracy 96.54% and validation accuracy 95.86%” (Afrin et al, 2019), with similar high results being achieved by Aptoula, Ghazi, and Yanikoglu (2017) who conducted comparative research, where they used a CNN with 3 different deep learning architectures, these being GoogLeNet, AlexNet, and VGGNet, in which their best case achieved an “overall accuracy of 80% on the validation set” (Aptoula, Ghazi, and Yanikoglu, 2017).

In addition to this Gajjar et al (2021), achieved a high accuracy when using a CNN to identify different plants to determine not only their identity but also to conclude whether the plant was healthy or diseased and if so what plant disease that might be, “that the proposed CNN architecture performs well in classification of diseases from leaves, giving an accuracy around 96.88%” (Gajjar et al, 2021).

An interesting point to note is that there are other ways to identify plant leaves other than through the shape and colour of leaves, which can be effectively picked out by a CNN, as noted by Chan, Lee, Remagnino, and Wilkin (2015) who implemented a CNN to identify plants not only based on leaf shape but also based on the venation structure, referring to the vein structure inside the leaf, of the leaf itself, achieving a high accuracy as a result. “Moreover, we demonstrated that venation structure is an important feature to identify different plant species with performance of 99.5%, outperforming conventional solutions.” (Chan, Lee, Remagnino, and Wilkin, 2015). This is significant as this provides a new area of exploration for plant identification, whilst also providing a potential method to achieve high levels of accuracy with a CNN, as, with this method, Chan, Lee, Remagnino, and Wilkin CNN have outperformed every AI system discussed previously.

However, using a full CNN for both feature extraction and identification might be unnecessary as concluded by Chao, Li, and Nie (2020), who proposed a potential implementation where a shallow CNN consisting of four convolutional layers and two pooling layers, for feature extraction and then utilised an SMV to achieve effective identification. They proposed this due to one of the primary weaknesses of CNNs, that they require a large number of parameters and layers to function effectively, as they stated, “The popular deep learning models require lots of parameters and layers to enhance their learning ability, consuming amount of computing resources.” (Chao, Li, and Nie, 2020). This means any way of effectively scaling a CNN down whilst still gaining the benefits of using it would be useful, as shown by their shallow implementation, combined with a lighter weight SVM to do the plant identification after the CNN handles the feature extraction.

### 2.2.3 Evaluation of SVMs

As concluded by the analysis of the comparative experiments undertaken by Balasaravanan, Priya, and Thanamani (2012) comparing the effectiveness of k-NN and SVM for image classification, SVM achieves a high level of accuracy when classifying different plants, with them stating, “The accuracy obtained by SVM in flavia dataset is 94.5%” (Balasaravanan, Priya and Thanamani,2012). This result was improved upon with their real-world testing “In the case of the real dataset, the accuracy of k-NN is 81.3% and the accuracy of proposed SVM classification approach is 96.8%” (Balasaravanan, Priya, and Thanamani, 2012). This high level of accuracy was also concluded by Arora and Kour (2019) who used a SMV that was optimised with particle swarm optimisation, achieving an average result of “classification accuracy = 95.23” (Arora and Kour, 2019) in comparison to other algorithms that were implemented, including an ANN optimised with a Genetic algorithm which only achieved an average accuracy of “85.42”(Arora and Kour, 2019).

However, SVMs do come with disadvantages, as concluded by Patle and Prajapati (2010) the effectiveness of a SVM is highly dependent on the kernel used, “The choice of kernel is an important issue in the SVM algorithm, and the performance of SVM largely depends on the kernel. As per our knowledge, no general rule is available as to which kernel should be used.” (Patle and Prajapati, 2010). This is an issue as there is no set method of determining the most effective kernel for a given problem without testing each potential kernel, making the process of implementing a SVM time consuming, most of which will be spent on testing kernels. Furthermore, SVM does not scale well with larger data sets, as concluded by Akata, Harchaoui, Perronnin, and Schmid (2015) who stated that “one of the limitations of nonlinear SVM classifiers is that they do not scale well with the number of training samples” (Akata, Harchaoui, Perronnin, and Schmid, 2014). Furthermore, SVMs have an issue with the lack of overall transparency in their results, as stated by Abdullah et al (2014), who stated that “SVMs have also some disadvantages. A common one is the lack of transparency in results.”(Abdullah et al, 2014) making calibrating and fine tuning a SVM difficult.

### 2.2.4 Evaluation of AI Methods Conclusion

It can be concluded that for plant classification a CNN is the most suited, with the ability to achieve the highest level of accuracy based on the research conducted, in comparison to a standard ANN and an SVM. Furthermore, the benefit of the CNN being able to do both feature extraction and image classification would allow for a shorter implementation in comparison to if the other methods were used which require a separate algorithm to undertake feature extraction.

## 2.3 Optimisation Methods for a CNN

To reduce the error in the CNN an optimisation algorithm is needed. It was determined by Rao and Vani (2019) in their study where they implemented and compared 7 identical CNNs with different optimisation algorithms, trained on a dataset consisting of different types of Indian Pines. These algorithms include, “Stochastic Gradient Descent (SGD), RMSProp, Adam, Adamax, Adagrad, Adadelta, and Nadam” (Rao and Vani, 2019). From their experiments, it was concluded that for a CNN, the Adamax optimisation algorithm was the most effective, outperforming all other implemented optimisation algorithms, with them stating that the “Adamax optimizer has outperformed the remaining with an accuracy of 99.58%. (Rao and Vani, 2019). However, in another comparative study of the best optimisation algorithm for a CNN, in this case for the image processing of brain tumours, undertaken by Arshid et al (2020) it was concluded that ADAM was the more effective optimisation algorithm, with it achieving an accuracy rate of “0.99” (Arshid et al, 2020) after 300 epochs outperforming their implementation of Adamax which achieved “0.96” (Arshid et al, 2020). This suggests a comparative implementation should be made, testing both the Adam and Adamax optimisation algorithm to determine which will be most suitable for the final version of the houseplant CNN.

## 2.4 Integrating Neural Networks with IoT/Mobile Devices

Due to how the system is required to function and to ensure the user can quickly and easily identify a given plant, having the CNN effectively work on less powerful devices, such as smartphones, is necessary. As concluded by Alsing (2018) in their thesis, one potential method of integrating this CNN is through the use of TensorFlow, a software library developed by Google to give the user extensive access to machine learning algorithms, and the use of its alternative version TensorFlow Lite, as stated by Alsing “TFL is the evolution of TFM, which already supports deployment on mobile and embedded devices” (Alsing, 2018) making it ideal for the mobile application integration of the required CNN model. This was also concluded by Karthikeyan (2018), in their book, where they stated that “TensorFlow Lite is a lightweight, energy-and memory-efficient framework that will run on embedded smaller-form factor devices.” (Karthikeyan, 2018) making it a suitable tool for use of CNN integration on mobile phones.

Furthermore, as concluded by Okamoto, Tanno, and Yanai (2016) in their paper where they examined different CNN architectures to determine which is most suitable for mobile implementation, for android devices a CNN consisting of a NEON SIMD instruction set was the most effective, having the quickest response time as well as concluding that for iOS devices the CNN architecture consisting of a BLAS library was the most effective, “As results, it has been revealed that BLAS is better for iOS, while NEON is better for Android” (Okamoto, Tanno, and Yanai, 2016). Furthermore, they also conclude that reducing the size of input image was an effective method of lowering processing time, up to a point, as they stated “Until 180x180, reducing the size of an input image when using the CNN is effective and easy way to adjust the trade-off between accuracy and processing time”(Okamoto, Tanno, and Yanai, 2016).

## 2.5 Ethical Considerations

This project does not use any personal or sensitive information so this project from an ethical point of view being very low risk.

# 3 Requirement Analysis

## 3.1 Introduction

The following consists of the requirement analysis of the system, this includes user stories as well as the functional and non-functional requirements alongside acceptance testing.

### 3.1.1 Priority System

Throughout this section, the MoSCoW priority system will be used to identify the necessity of each requirement. As stated by Garleanu, Mărzan, Paul, Spiru, and Velciu, “MoSCoW stands for must, should, could and have requirements to accomplish business needs.” (Garleanu et al, 2019) and is primarily used to conclude between designers and developers what should be developed, which must be prioritised as well as what features are not viable. This method is ideal for this project as with a restrictive time limit and the necessity to balance this work alongside other modules, compromises on scope and the overall functionality must be made.

### 3.1.2 Justification of Requirements

Throughout this section, any design choices made, including system functionality, requirements and their priority will be justified by stating the source of these requirements.

## 3.2 User Stories

The following is a sample of the user stories. To see them all, see Appendix 10.1.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | As a… | I want… | So that… | MoSCoW |
| US.AI1 | User | A method of identifying what my house plant is | I can identify unknown plants that I have in my possession | M |
| US.AI2 | User | To know the name of the plant that I am having identified | I know what the unknown plant is called. | M |
| US.AI3 | User | To be able to identify lots of different house plants | I can identify many different houseplants | M |
| US.AI4 | User | The plant identification to be accurate. | I can ensure that the plant is not misidentified | M |
| US.AI5 | User | This plant identification to be done quickly. | I can easily determine what my unknown plant is promptly. | M |

Table 1: User Stories

## 3.3 Functional Requirements

The following are a sample of the functional requirements for the neural network, the user interface, and the database. To view them all, see Appendix 10.1.2.

### 3.3.1 Functional Requirements of the Neural Network, User Interface, and Database

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirements | Priority (MoSCow) | Source/ Justification |
| AI1 | The AI system will need to identify images of house plants | M | Core functionality |
| AI2 | The AI system will need to support different species of house plant | M | Core functionality |
| AI3 | The AI system must be able to be integrated into a user interface | M | Core functionality |
| AI4 | The AI system must return the name of the houseplant identified | M | Core functionality |
| AI5 | The AI system must be able to handle invalid images | M | Core functionality |
| UI1 | The system will require a front-end user interface | M | Core functionality |
| UI2 | The user interface will need to interface with the neural network | M | Core functionality |
| UI3 | The user interface will need to interface with a database | M | Core functionality |
| UI4 | The user interface will need to pull relevant plant information from the database, based on the results of the neural network | M | Core functionality |
| UI5 | The User interface must allow the user to upload an image to the CNN | M | Core functionality |
| DB1 | The database must be able to store data | M | Core functionality |
| DB2 | The database will need a table to store care information about plants | M | Core functionality |
| DB3 | The databases plant information table will need a column to store plant names | M | Core functionality |
| DB4 | The databases plant information table will need a column to store basic plant descriptions | M | Core functionality |
| DB5 | The databases plant information table will need a column to store a plants ideal light level | M | Core functionality |

Table 2: A Sample of the Functional Requirements

## 3.4 Non-Functional Requirements

The following is a sample of non-functional requirements for each component of the system. To view the others, see Appendix 10.1.5

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirements | Priority (MoSCoW) | Source/ Justification |
| NF.AI1 | The CNN must have a minimum accuracy of 95% on training data | M | Core functionality & Literature review: Arfin et al (2019), Aptoula, Ghazi, and Yanikoglu (2017), Gajjar et al (2021), and Chan et al (2015) |
| NF.AI2 | The CNN must have a minimum accuracy of 90% on validation data | M | Core functionality & Literature review: Arfin et al (2019), Aptoula, Ghazi, and Yanikoglu (2017), Gajjar et al (2021), and Chan et al (2015) |
| NF.AI3 | The CNN must have a minimum accuracy of 90% on test data | M | Core functionality & Literature review: Arfin et al (2019), Aptoula, Ghazi, and Yanikoglu (2017), Gajjar et al (2021), and Chan et al (2015) |
| NF. AI 4 | The CNN once integrated with the mobile application, must have a response time of less than 1 second to identify a plant | M | Core functionality & Literature review: Okamoto, Tanno, and Yanai (2016) |
| NF. AI 5 | The CNN must be able to identify at least 10 different species of common house plants | M | Core functionality |

Table 3: A Sample of the Non-Functional Requirements

## 3.5 Acceptance testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Which requirements are being tested? | How they are to be tested | Pre-requisites | Expected output |
| AT1 | AI1, AI2, AI3, AI4, AI7, AI9, AI10, UI1, UI2, UI3, UI4, UI5, UI7, UI8, UI10, DB1, DB2, DB3, DB4, DB5, DB6, DB7, DB8, DB9, DB10 | Upload an image, through the camera, of a plant for the CNN to identify | The system must be running | The system will return the label of the plant that it deems the image to be, as well as display the image of the houseplant. All relevant care information for the plant that matches the label returned by the CNN will also be displayed |
| AT2 | AI6, UI6, UI9 | Upload a poor image of a houseplant after a houseplant image has already been submitted | The system is running, and an image has already been uploaded to the system | The system will clear the previous plant photo and information and attempt to identify the new image to the best of its ability, displaying the new plant image and its information, determined by the label. |
| AT3 | AI5 | Upload a random image of something that is not a house plant to the system. | The system must be running | The CNN will attempt to label the image the best it can, returning the label with the highest probability as determined by the CNN, at which point the image and the relevant plant data for the label are displayed |
| AT4 | AI8 | Two identical CNN models will be created with two different optimisation algorithms | N/A | The two CNNs will be trained, the method with the highest accuracy and validation accuracy will be chosen and integrated into the application |

Table 4: Acceptance Tests

# 4 Methodology

A choice of methodology for a project must consider the following factors,

* Firstly, the project is being undertaken by one person, resulting in all methodology roles being undertaken by one person.
* Secondly, the development process must be flexible to allow for both the completion of this project as well as work for other modules.
* Thirdly, each component of the methodology must be able to be taken in parallel, i.e., the research for the CNN must be able to synchronise closely with the development of the CNN as well as the testing of the CNN.
* Finally, the methodology must allow for adaptive code implementation alongside rigorous documentation.

As stated by Butler and Vijayasarathy (2016) there are three schools of thought in software development methodology, these include the traditional linear software development approach, known as waterfall, the more modern adaptive approach, known as agile, and finally hybrid versions of these approaches.

## 4.1 Chosen Methodology

A hybrid approach will be taken, consisting of Scrum with a primary focus on documentation at the end of each sprint, like Waterfall, as well as the implementation of core functionality in a sequential and prototype driven fashion, with each component of the development and the documentation being broken up into core phases, which are them broken down into sprints.

## 4.2 Project Plan and Timeline

The following Gantt chart and hierarchical breakdown of the system were produced to show the estimated time frame and structure of the project.

### A picture containing bar chart Description automatically generated4.2.1 Gantt chart

Figure : Project Gantt Chart

As shown in **Figure 1**, due to the workload and the component like nature of the system, some phases of the project are intended to run in parallel to each other, to allow for effective time management and to allow for effective integration and testing of each subsystem of the artifact, as the parallel development allows for greater focus to be placed on the smooth implementation and integration of each component.

### Diagram Description automatically generated4.2.2 Hierarchical Breakdown

Figure : Project Hierarchical Breakdown

As shown in **Figure 2**, the project will be broken down into five phases, this being the research phase which will be focused on documentation, the design phase, which will be focused on the design of each component of the system as well as the overall structure of the system. The software development phase consists of four components with each of the three main subsystems of the artifact having their subcomponents that must be implemented before they are finally integrated into the artifact. The testing phase is focused on ensuring the artifact functions as intended and meets all requirements that have been set, if it does not, this is where the project can safely loop back to any of the previous stages, ensuring the agile methodology is adhered to. Finally, the project closure phase will be focused on the final project evaluation, identifying what was achieved in the project, what could be improved as well as where the artifact and the project can be taken from here.

# 5 Design

The following section consists of the design of the artifact, this has been separated into the design of the overall system, the design of the CNN, the design of the mobile application, and the design of the database, with justifications for each design choice.

## 5.1 System Architecture

Diagram

Description automatically generated

Figure : IdentiFlora System Architecture

**Figure 3** shows an overview of the overall system, the system is intended to be structured to ensure low coupling between components, as shown above with by the TensorFlow Lite model and the database having no direct interaction between each other, instead, using the mobile application as the central point of communication and interaction, resulting in less redundant interactions between modules in the system. Furthermore, the system has been designed to have a high level of cohesion, as demonstrated by the strong centralisation of data and processing onto the mobile interface. This is to ensure reduced complexity in the system, whilst also making the development of the system and debugging the system far easier as this reduces potential points of failure and, when combined with the low coupling, makes isolating points of failure straightforward.

## 5.2 Design of the Neural Network - Prototype CNN

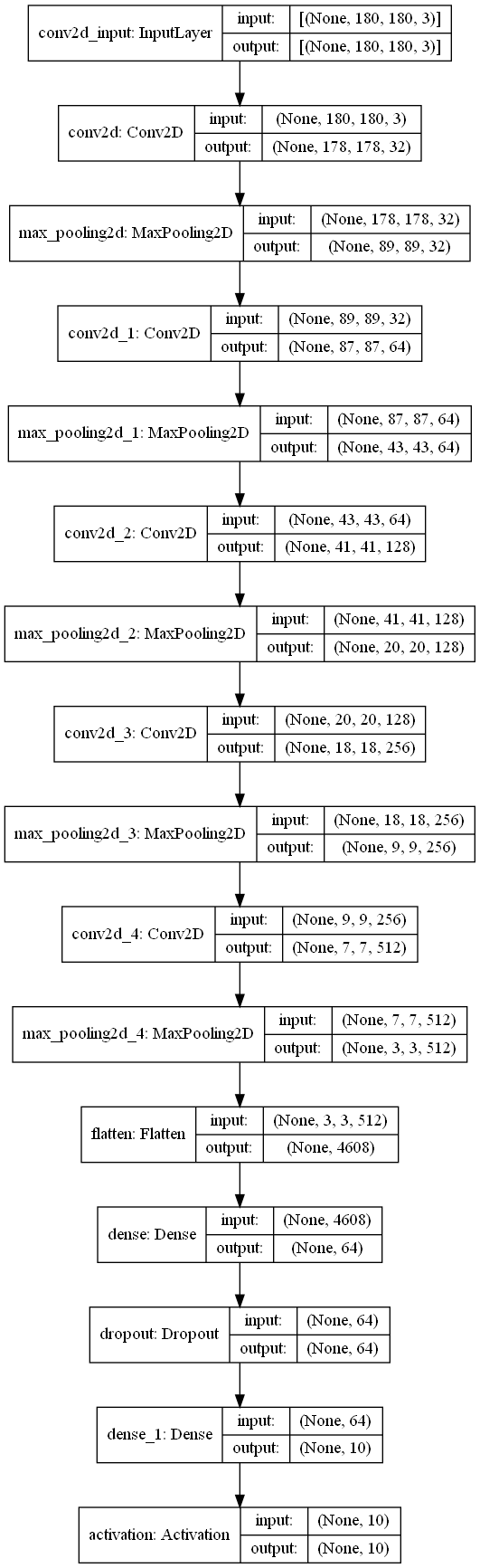


Figure : Initial Protoype CNN structure

**Figure 4** displays the intended structure of the CNN, this will consist of a sequential model of five convolutional layers using “rulu” activation, with the initial input layer reshaping images into a 180x180 pixel format, with each following convolutional layer doing feature extraction. The final layer will consist of a flattening layer that will reduce the input data from the previous neurons into a single dimension, this is to allow for more effective classification based on feature extraction from the previous 5 convolutional layers. This is then to be followed by a dense layer using “rulu” activation where all previous neurons will be linked to one dense neuron, this is to be followed by a dropout layer to assist in preventing overfitting. This will then be followed by a final dense layer using “softmax” activation which allows for effective classification of images where there are more than two labels (i.e., plant species) that can be identified.

For the choice in optimisation algorithms, the comparison between “adam” and “adamax” will take place in the development phase, using the above CNN structure to conclude which method would result in overall higher accuracy, validation accuracy, and testing accuracy.

The model is then intended to be trained over a minimum of 20 epochs, with an 80/20 split of image data, with 80% becoming training data, and the remaining 20% being split again, with 10% of the overall data becoming validation data and the remaining overall 10% becoming testing data.

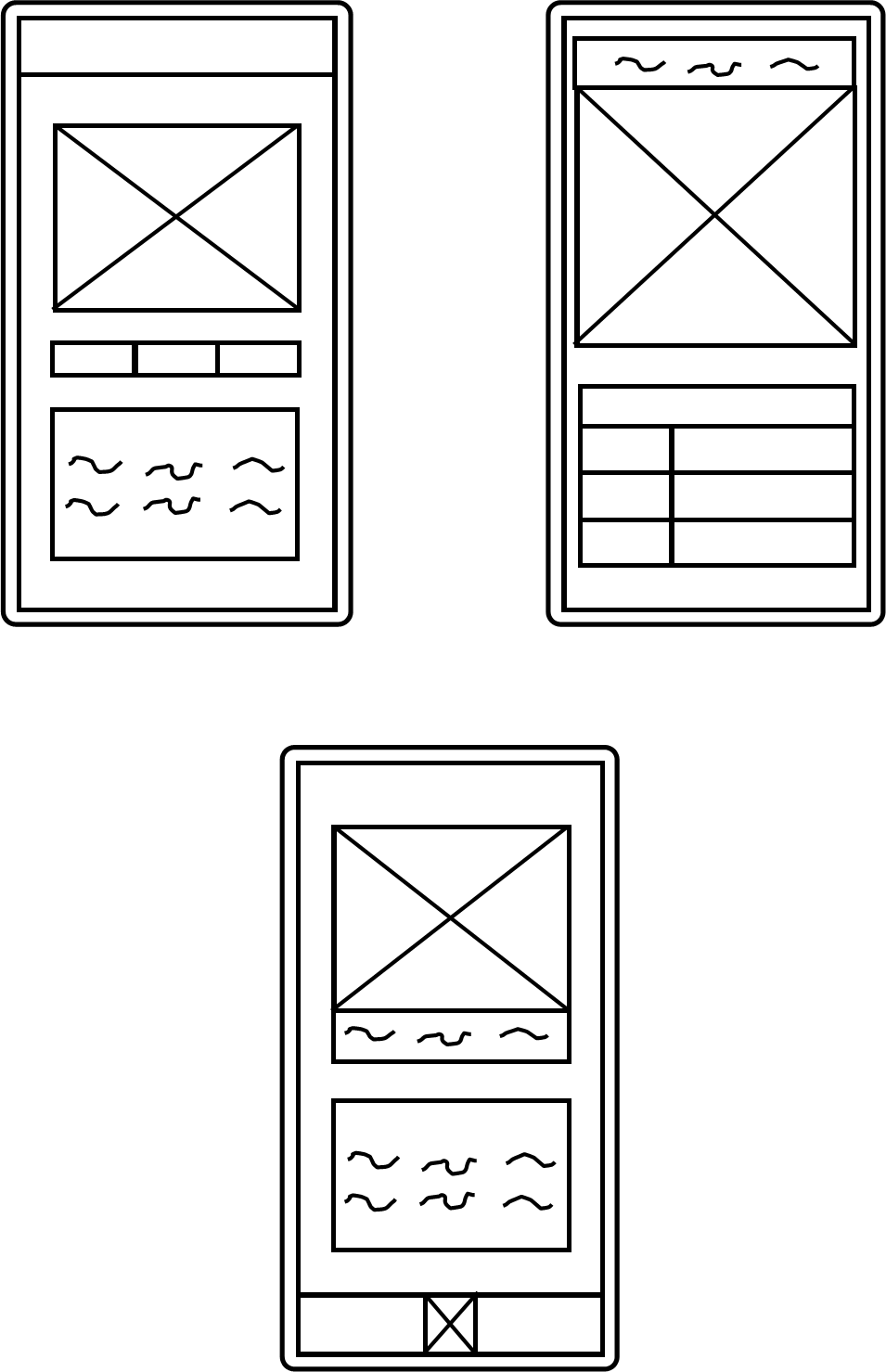
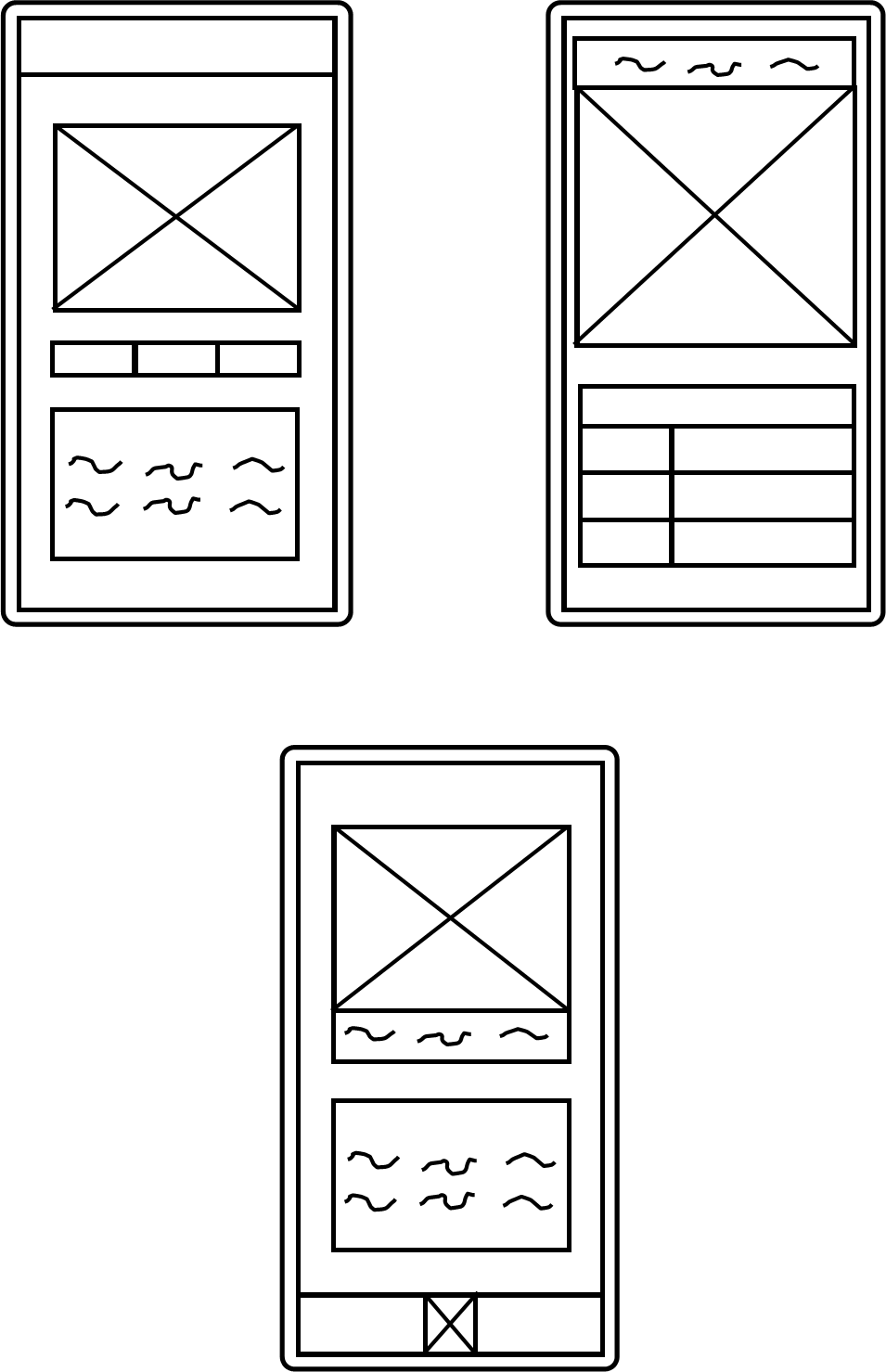
Due to the nature of neural networks, the exact numbers, such as epochs, and activation algorithms are subject to change to allow for effective fine-tuning of the CNN.

## 5.3 Design of the Mobile Interface

### 5.3.1 Wireframes

#### 5.3.1.1 Initial Mock-ups

Figure : Initial wireframe mockups



As shown in **Figure 5**, the application interface will consist of three components, these include the central image, which is intended to display the user’s plant that they are identifying, the table/text layout that will display the care information for the identified house plant, and finally, the interactions layer, which will be how the user uploads an image of their house plant to the system. Image uploading could be done in two possible ways, the required option is to be able to upload the image by letting the user take the image through the application using their phone's camera, this would be done by tapping a button, consisting of the central image. Furthermore, it would be beneficial to have the option to upload an image from the user's locally stored files.

#### 5.3.1.2 Final Version

A picture containing chart

Description automatically generated

Figure : Final version of the mobile interface wireframe

As shown in **Figure 6**, the application is intended to consist of two primary screens, a splash screen when the user launches the application and the main screen, where the user can take a photo of the plant they want identified and upload it to the system using the image button, in addition, the care information of that plant then be displayed in the care information table.

### 5.3.2 Colour Schemes

Proposed Scheme 1:

Figure : Colour scheme 1

Max Green

Hex Code : #4c9a2a

RGB (76, 154, 42)

Purpose : Banners and header section

Very Deep Spring Green

Hex Code : #011910

RGB (1, 25, 16)

Purpose: Background

White

Hex code: FFFFFFFF

RGB (255,255,255)

Purpose : Text

The above proposed colour scheme is designed around the idea of it being used as a dark mode for the application, allowing the user to customise the experience whilst also assisting those with visual impairments to interact with the application more effectively.

Proposed Scheme 2:

Figure : Colour scheme 2

Max Green

Hex Code : #4c9a2a

RGB (76, 154, 42)

Purpose : Banners and header section

Black

Hex Code : #000000

RGB (0, 0, 0)

Purpose : Text

White

Hex code: FFFFFFFF

RGB (255,255,255)

Purpose: Background

The second proposed colour scheme would be used for a light mode in the application, with bright contrasting colours, whilst keeping the plant aesthetic, makes this suitable for those with the most common forms of colour blindness assisting in making the application more accessible as well as assisting in making the user experience more pleasant and customisable.

### 5.3.3 Composites

**Graphical user interface

Description automatically generatedFigure 9** consists of the composites for the mobile application. These composites are to demonstrate both the layout and colour scheme of the application.

Figure : Mobile Interface Composites

The application is intended to work with minimal input from the user, once the photo is uploaded to the application, all the work done in the system is done by the CNN and the database, which is done automatically, resulting in only three interactions in the core interaction loop of the application, consisting of tapping the central icon, taking a photo and then confirming the submission of that photo, which can then be repeated for each plant they want identified. This minimal interaction loop is to ensure the user can obtain the information they want as quickly making the use of the application seamless and efficient.

### 5.3.4 Assets

The following section consists of design work done for each asset of the application interface.

#### 5.3.4.1 Logo

Logo

Description automatically generatedLogo Initial digital mock-ups:

A picture containing diagram

Description automatically generated

Figure 10 : Initial digital mock-ups of potential logos

The initial logo designs consist of simplistic stylised representations of a cylinder snake plant, assisting in demonstrating the core premise of the application in a fun and stylised manner. The second design expands upon this idea with the addition of the magnifying glass, including a small zoom-in effect that makes the premise of the application which is identifying plants more explicit to the user at a passing glance.

Logo Design 1:

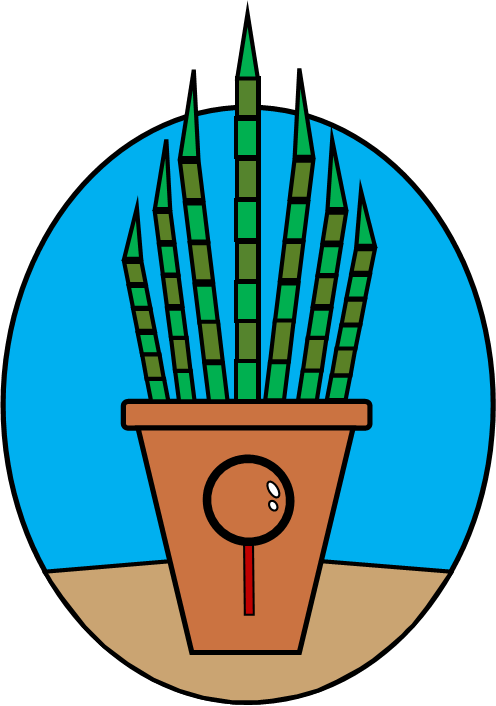


Figure : Logo Design 1

**Figure 11** has expanded upon the original design by firstly the addition of the small magnify glass on the plant’s pot, to better signify the purpose of the application. Furthermore, the logos background has been made more oblong to guide the users focus to the core image of the cylinder snake plant. Finally, colour has been added assisting in making the logo visually interesting as well as assisting the user in better identifying the applications purpose from a glance.

Logo Design 2:



Figure : Logo Design 2

**Figure 12s** main improvement is the addition of more colour to the image, making it look more vibrant than its original counterpart. This logo benefits from a squarer design making it more suitable as a standard icon, due to the way both Android and iOS devices display app icons as small squares with circular edges.

## 5.4 Design of the Database

Table

Description automatically generated

Figure : The Plant Care inforatmion Database Structure Diagram

As shown in **Figure 13** the database will consist of a small, one-table SQLite database, integrated internally into the mobile application. The rationale behind this decision is that the database is only intended to be read from and never written to by the users so a consistent internal database would be highly beneficial as storage becomes less of an issue and this allows for greater speeds of access in comparison to an eternally hosted server, as well as circumvents the need for the application to have an internet connection, allowing for plant identification no matter where the user is. Furthermore, for this database to require new entries, the CNN would need to be able to identify a new houseplant, so updating the database is only relevant when a major change to the AI model is made, which would require a full software update of the application, so the database would therefore need to be updated in tandem.

# 6. Implementation & Testing

## 6.1 CNN Implementation

### 6.1.1 Overview

In this stage of the project, the CNN component of the system was created, this was planned to be broken down into five sprints. These sprints consisted of obtaining the datasets for training, image pre-processing, the initial creation of the CNN, the evaluation of optimisation algorithms, AI fine-tuning, and optimisation.

### 6.1.2 Datasets

#### 6.1.2.1 Pre-existing Datasets

The initial intent was to use pre-existing datasets to train the CNN model however, upon further research it was concluded that this was not viable, as after an in-depth search only one dataset consisting of 4 different species of houseplant was found. This lack of pre-existing specialised datasets meant that a choice had to be made. The first option was to expand the scope of the project to include other plants, such as different types of wild plants and non-indoor plants, which went against the purpose of the project. The second option was to create the datasets needed, which came with its own set of challenges, firstly these images would need to be gathered, a time-consuming task that to do effectively required access to many different houseplants of the same species, whilst ensuring that variants of species are also covered to ensure that these variants plants can be identified successfully.

To ensure the purpose of the project was adhered to as well as allow for experience to be gained, it was decided to create the remaining houseplant datasets.

#### 6.1.2.2 Creating Datasets

To obtain images to create these new datasets, images of houseplants from the author's private collection were taken, which laid the groundwork for the different species of houseplants the CNN would be able to identify. However, using only these plants would limit the effectiveness of the AI due to plant variations in the same species not being accounted for. This meant that using only the collection of houseplants the author had was not ideal, as using the same specimen of plant multiple times would significantly decrease the accuracy of the AI when it would attempt to identify other plants of the same species, due to the lack of variety that it would have been trained on. To supplement this, excursions to local plant nurseries and garden centres were made to obtain additional samples to ensure the variety of plants of the same species in the final dataset was extensive enough to be effective.

At the end of this process, there were 4 pre-existing plant species datasets all containing 150 images per species, and 7 handmade plant species datasets containing 100 images per species of plant, resulting in the final version of the CNN for this project being able to identify 11 different plants.

### 6.1.3 Image Pre-Processing

Firstly, the datasets had to be balanced with each species of plant having the same number of images. This is to assist in making the CNN more effective at identifying each plant to an equal amount, giving it an overall higher accuracy as, as concluded by Lalithnarayan (2020), an imbalanced dataset results in lower overall accuracy and the CNN having a natural bias toward predicting the input classes with the higher number of images. The result was all plant classes being cut down to 100 images each.

In addition, a python program called CNNImagePreprocessor.py was created, the purpose of this program was to reformat all training images to a 180 by 180-pixel format, label them accordingly and convert the image data into a 3D NumPy array to store coloured image data and a 2D array to store the corresponding labels of the plant. This was done to allow for both image data and colour data to be considered when this data is being used to train the CNN, to allow for more effective feature extraction, and to achieve better results when the AI undertakes real word plant identification.

A further program was intended to be created called CNNImageNoise.py that would do minor image manipulation on training images. This was designed as minor image manipulation, including tilting images at different angles, and the process of salt and peppering to assist in making the AI handle noisy input effectively. However, this program was later shelved halfway through development due to issues with hardware and training time, meaning increasing the number of images in the dataset was not viable.

### 6.1.4 Optimisation Algorithm Comparison

A rudimentary CNN, see **Figure 14**, was created in python using the TensorFlow library, in the PrototypeCNN.py file, to allow for the testing of the two optimisation algorithms. Each CNN was trained on the 4 pre-existing datasets, 100 images each for 20 epochs each.

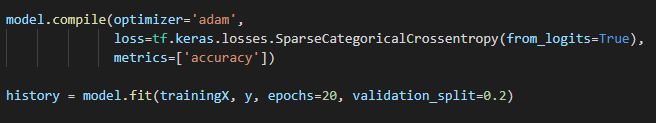
Text

Description automatically generated

Figure : Prototype CNN TensorFlow Model for optimisation algorithm testing

#### 6.1.4.1 Adam Optimisation

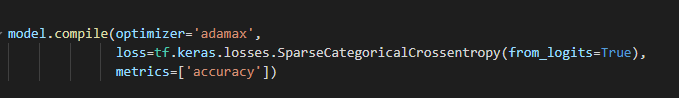
Figure : Model compiler using Adam optimisation and results



After training, the prototype CNN using “Adam” optimisation achieved an overall accuracy of 78.75% with a validation accuracy of 77.5%.

#### 6.1.4.2 Adamax Optimisation

Figure : Model compiler using Adamax optimisation and results



After training, the prototype CNN using “Adamax” optimisation achieved an overall accuracy of 80% with a validation accuracy of 82.5%.

#### 6.1.4.3 Optimisation Algorithm Comparison Conclusion

As shown above, the “Adamax” optimiser outperforms the “Adam” optimiser for this specific use case, which is also reinforced by the literature, which states that whilst “Adam” is an effective general optimisation algorithm, “Adamax” is capable of outperforming it, especially with models that have high levels of embedding, where complex and large data inputs consisting of high-resolution images which also include pixel colour data, have been translated into a low dimensional space for training purposes, in this case, flattened from a 3D array to a 2D array. This means “Adamax” would have a slight advantage in the current use case, as shown by the comparison between the two optimisation algorithms.

### 6.1.5 Initial CNN Training

In this sprint the CNN architecture was improved upon and refined, as well as the initial AI training took place.

The plan was to train the CNN as a TensorFlow model, then design a python program that would then convert that model into a TensorFlow Lite model. To achieve this a python program called tensorFlowToTensflowLiteConverter.py was created, this loaded in a .h5 TensorFlow model and then converted it into a .tflite model using the TensorFlow tf.lite.TFLiteConverter.from\_keras\_model() function.

This method whilst successful came with drawbacks, for example, it was found that the use of this model would occasionally cause the application to crash. The solution to this was to build the model natively for TensorFlow Lite, this was achieved through the creation of tensorFlowLiteModelGenerator.py, this program combined both the image pre-processing of the CNNImagePreprocessor.py and the model generating capabilities of PrototypeCNN.py, outputting a trained TensorFlow Lite model alongside the relevant labels of each species of plant. This was achieved using the TensorFlow and the TensorFlow Lite Model Maker python libraries, which assisted in the simplification of model creation using automatic image pre-processing and the ability to use pre-existing as well as custom model architecture. From this, successful stable TensorFlow Lite models were created, allowing for the CNN model development to move forward.

Attempt 1: CNN Trained on All Plants

Text

Description automatically generated

Figure : Results of training the first TensorFlow Lite implementation of the CNN

As shown in **Figure 17**, after training for 10 epochs, trained on an 80/20 split between training data and validation and testing data, the CNN achieved an accuracy of 95.83%. However, when compared to testing data, the model only achieved an accuracy of 80%, which does not meet the required minimum accuracy for testing and validation data, as stated in the requirements.

Attempt 2: CNN Trained with More Validation Data and More Epochs



Figure : Results of training the second TensorFlow Lite implementation of the CNN plant classification AI

As shown in **Figure 18**, training the AI with 15 epochs, and adding more validation data, increasing the split to 75% training data, 10% validation data, and 5% testing data. The model achieved a greater training accuracy of 98.56%, validation accuracy of 96.36%, and a testing accuracy of 92.73%, meeting all requirements for model accuracy, and was chosen for later integration into the mobile application.

### 6.1.6 Final Product

The final product consists of a TensorFlow lite CNN model called IdentiFloraCNN.tflite that can identify 11 different species of houseplant.

### 6.1.7 Problems and Challenges with the Development of the CNN

One problem with this section of the project was that training times for the CNN, even with a small number of epochs, were substantial. This is primarily due to the insufficient hardware available for training, as the CNN was being trained on a laptop with Intel i5-8250U CPU @ 1.60GHz, 1800 Mhz, 4 Core(s), 8 Logical Processor(s), an NVidia GTX 1050 with 2GB of Vram, and 8GB of 2400Mhz DDR4 ram. At the beginning of the project this was not the plan, as the AI component was intended to be trained on a desktop PC with an Intel i7 6700K CPU @ 4 GHz, 4 Core, 8 Threads, an NVidia GTX 1060ti with 6GB of Vram and 16GB of DDR4 3000 MHz, however, due to time constraints and lack of access to the PC, this was not possible. There were two potential solutions, the first one being to limit the amount of training data, this would speed up the training time significantly at the cost of making the AI system less effective. The second potential solution was to use Google Colab, this would allow for the training to be done on external Google servers, however, to get access to sufficient hardware and to prevent the system from timing out it would require a monthly subscription.

It was decided that cutting down the number of images for each specimen in the system was the most viable solution given the lack of hardware.

## 6.2 Mobile Application Implementation

### 6.2.1 Overview

In this phase, the mobile application interface was created. The first sprint involved the initial creation of the base application, which consisted of the creation of the layout and visual components of the application following the composites and wireframes. The second sprint consisted of implementing the camera functionality ensuring input images were displayed as intended.

### 6.2.2 Creating the Mobile Application

This sprint consisted of the creation of the user interface of the application, due to the extensive design work done in the design phase of the project, implementing the UI based on the composites and wireframes was a simple task. To achieve this, firstly the initial base application was created in Android Studio, and the primary activity screens were designed using the Android pallets provided, with the central image consisting of an imagebutton pallet to allow for later integration of camera functionality, the use of the tablelayout pallet to create the care information table, as well as the initial implementation of the dark mode colour scheme, the importation of the previously designed image assets, as well as the implementation of the IdentiFlora app icon, shown in **Figure 19.**

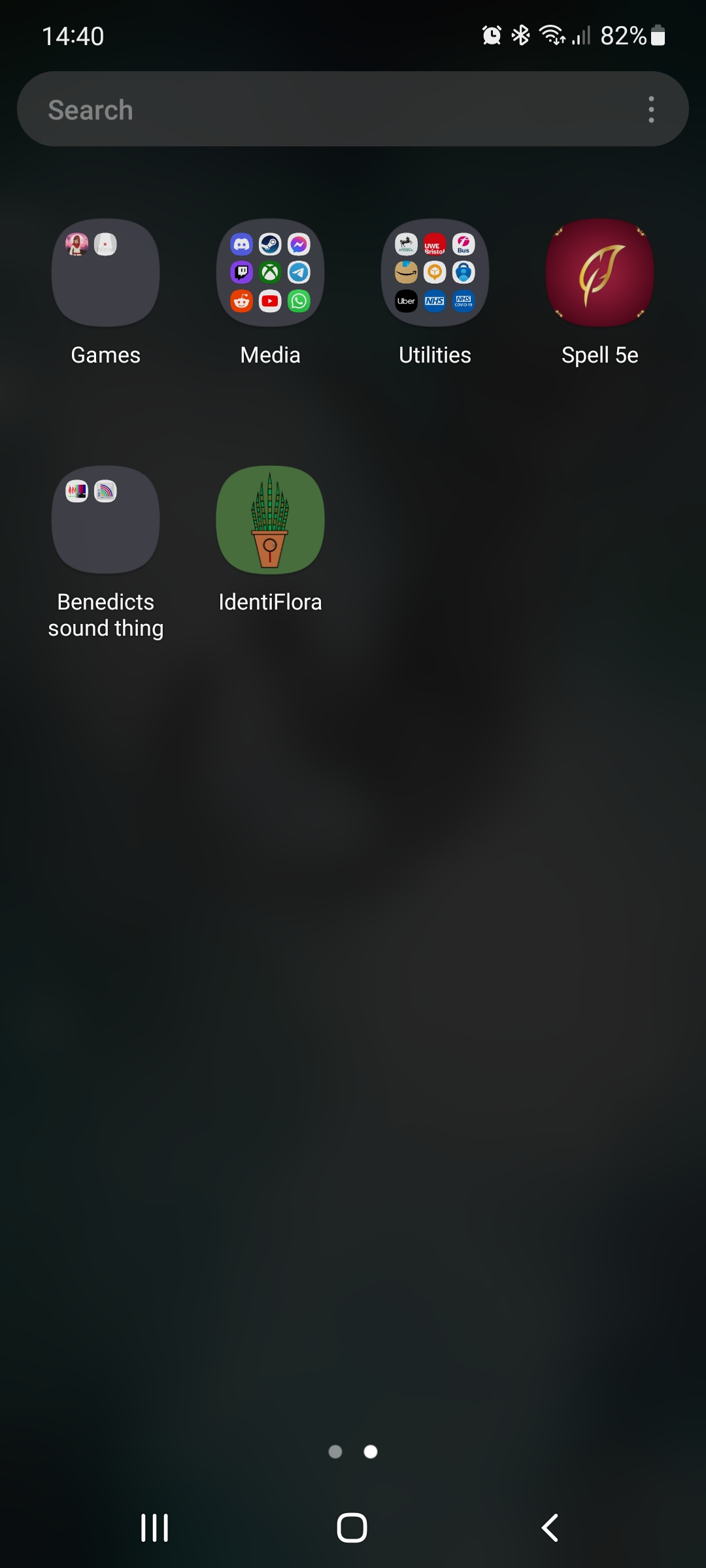


Figure : IdentiFlora App Icon

The completion of this sprint concluded with the successful implementation of the user interface, as shown in **Figure 20**.



Figure : Application interface as viewed in Android Studio .xml design editor

### 6.2.3 Camera Utilisation and Integration

This sprint involved the implementation of the camera functionality of the application. There were two possible implementations for this, one would be to call the camera activity, handing activity control over to the device’s camera application, then the camera application would directly pass the image data as an intent bundle from the camera and store it as a local variable in the program itself. This would be the simplest option and one of the recommended ones in the Android Studio documentation, however, this limits image file sizes to 1MB resulting in an overall poor quality looking image being displayed as well as results in worse CNN performance due to the noise in the image data. The second option would be to pass in a file location to the camera application when the camera activity is called, this would specify where the image should be stored locally, once the user has taken their image and it has been stored, the activity control would be passed back to the application, which can then retrieve the relevant image from the local file storage. Whilst more complicated to implement, requiring android application permissions to store external files on the user device and requiring the implementation of a file provider, it allows for higher resolution images with no imposed size limit to be rendered in the application as well as would provide higher quality images to the CNN model, resulting in an overall better user experience. **Figure 21** shows the code implemented to achieve this.

Text

Description automatically generated

Figure : Code used to open the default Android camera application

A picture containing text

Description automatically generatedAs shown, in **Figures 22, 24,** and **25**, this worked as intended with the application successfully capturing and displaying the image, however as discussed in the problems and challenges section, there were minor issues with displaying the image taken by the camera however, the issue was corrected.

Figure : Camera open through the application

### 6.2.4 Final Product

As shown in **Figure 23**, the development of the application user interface has been successful, ready for the final phase of AI and database integration.

Graphical user interface, website

Description automatically generated

Figure : Screenshot of the mobile application at the end of the mobile application development phase

### 6.2.5 Problems and Challenges with the Development of the Mobile Interface

Issue 1: Any image taken in portrait through the application is displayed incorrectly.

Evidence:

Graphical user interface, website

Description automatically generated

Figure : Screenshot of an image taken with the camera not displaying correctly

Solution: Reformat all inputted images by 90 degrees

Evidence:

Graphical user interface, website

Description automatically generated

Figure : Screenshot of the outcome of the initial fix

The issue with this solution: This has resulted in any images taken in landscape now no longer displaying at all.

Solution: The issue was with how Android restarts the application when transitioning the phone from portrait to landscape and vice versa. There were two possible solutions, the first and most complicated solution was to save the state of the application before the transition from portrait to landscape, this would be done by taking the current state of the activity running on the Android application and then saving it to be displayed once the rotation had taken place.

The second and chosen solution was to not allow the application to be able to display in any other manner than portrait. This solution was implemented however resulting in the same issue of images not being displayed properly. This was solved by using the EXIF tags of the taken image which included the angle of the phone when the image was taken, once this angle is known, the image can then be rotated accordingly and displayed in the intended manner, this was achieved with the code shown in ***Figure 26***.

Text

Description automatically generated

Figure : Code used to rotate the image to match the rotation of the device using EXIF image tag data

## 6.3 Database Implementation

### 6.3.1 Overview

This phase was focused on the creation of the database. This was split into two sprints, firstly the creation of the database, and the second sprint being the implementation of plant care data into the database.

### 6.3.2 Creating the SQLite Database

In this sprint, a simple python programme, CreatePlantDatabase.py, was created. This programme, using the sqlite3 python library was used to create the initial table in the database and populate each header in each column with the appropriate fields, these being the plant name (which acts as the primary key for the table), the plant's description, the plant's water requirements, the required light levels, the ideal temperature, the ideal humidity, how often the plant requires to be fertilized and the finally any unique care information about the plant.

### 6.3.3 Implementing Plant Data into the Database

Text

Description automatically generatedIn this sprint, the plant information for each species of plant the CNN could identify was added to the database. This was a simple process of adding each plant’s details manually to the database using the cursor.execute() command using the column structure implemented in the previous sprint.

Figure : Database creation code

### 6.3.4 Final Product

Graphical user interface, application, table, email

Description automatically generatedAs shown in **Figure 28**, the database has been successfully created containing all the relevant plant data.

Figure : Plant information and care database visualised using SQLite Viewer

### 6.3.5 Problems and Challenges with Building the Database

Due to the simplicity of this stage of the project, no issues were encountered.

## 6.4 Component Integration

### 6.4.1 Overview

This phase was focused on integrating each component into the mobile application. This was split up into two sprints, the first being the integration of the CNN and making it provide predictions based on input images. The second was the integration of the database and making the database provide the correct information based on the predictions made by the CNN.

### 6.4.2 Integrating the CNN into the Mobile Application

Text

Description automatically generatedThis sprint consisted of integrating the CNN. Since both Android and TensorFlow are owned by Google, AI integration into Android applications is encouraged and extensively supported by Android Studio and the Kotlin programming language, which made the integration of the CNN a simple process of importing the model using the provided importation tool. Then, as shown in **Figure 29**, the model could then be called and a new instance of it could be created, from here an image can be provided for identification. Identification was achieved using the probabilistic values provided by the CNN, with the species class with the highest probability of being present being selected as the plant in the image. The model could then be closed ensuring the stability of the application, and the relevant plant label could be returned allowing for care information for that plant to be retrieved from the database.

Figure : Code used to obtain the probabilistic result of which plant is present in the provided image

### 6.4.3 Integrating the Database into the Mobile Application

This sprint consisted of integrating the database and creating the corresponding database handler, named PlantDatabaseHandler.kl to handle calls to and from the database and involved taking the relevant data from the database based on the CNN prediction and displaying it. As discussed in the problems and challenges section, getting the application to open the SQLite database proved obtuse and not as simple as the Android Studio documentation would suggest, however, once the database was integrated into the application and readable, the matter of displaying the relevant information based on CNN outputs could be tackled. The following block of code, **Figure 30**, in tandem with the database handler was used to achieve this.

Text

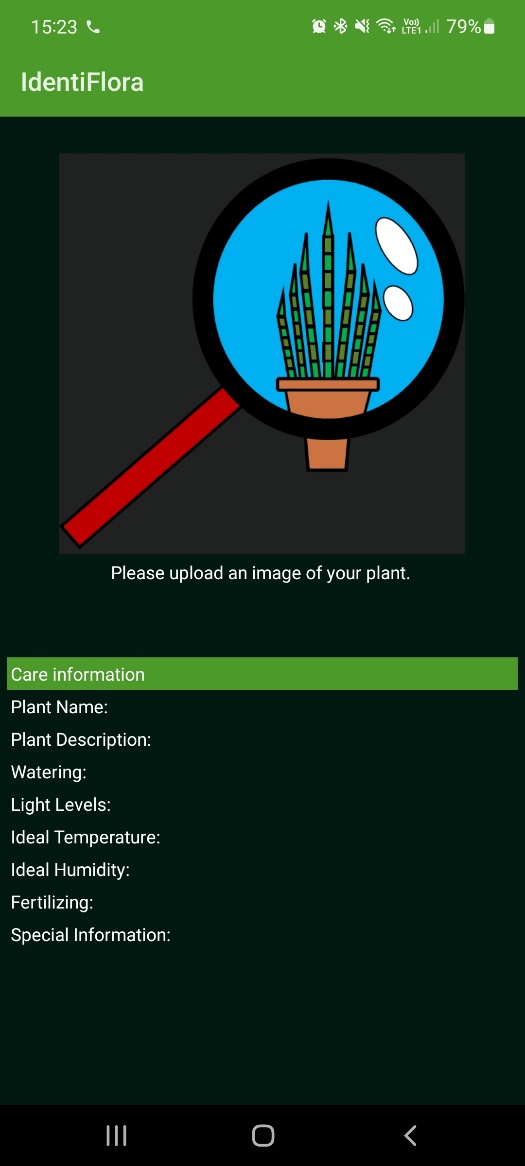
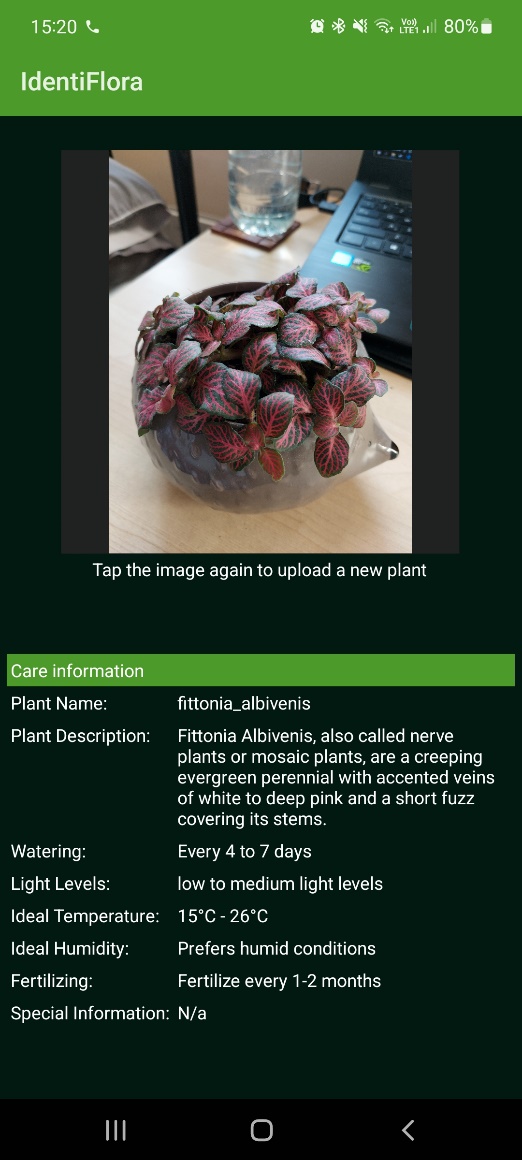
Description automatically generated

Figure : Code for displaying content in the table view from the database based on CNN output

### 6.4.4 Final Product

The artifact at the end of this phase represents the end of the development phase of the project, with the final product, as shown in **Figure 31**, being feature complete and ready to be tested.

Figure : Screenshot of the final application, shown here identifying a nerve plant



### 6.3.4 Problems and Challenges with Component Integration

Problem 1: The system cannot find the plant information table in the database

Text

Description automatically generated

Figure : Empty Database error message

Graphical user interface, application

Description automatically generated

Figure : Database made in python before porting over to the Android application

A picture containing application

Description automatically generated

Figure : The exported database that was being opened by the Android application

As shown in **Figures 33** and **34**, it was determined that in the process of the application opening the database from the assets folder the contents were being wiped. The initial conclusion was the database in the assets files was broken, however, it was later concluded that the database was fine, further supporting the argument that something was going wrong when the application opens the database, with data not being translated over properly.

Solution: Due to limitations with Android it cannot read from a database in the assets folder, a temporary copy of that database inside the code must be made, which is done using the following code in **Figure 35**.

Text

Description automatically generated

Figure : Code used to copy the database from the assets folder so it could then be used

## 6.5 System Testing

### 6.5.1 Acceptance Testing

This section will consist of the evaluation of whether the system has met the acceptance tests established in the requirements phase.

|  |  |  |
| --- | --- | --- |
| Acceptance Test Being tested | Pass or fail | Comments |
| AT1 | Pass |  |
| AT2 | Pass |  |
| AT3 | Pass |  |
| AT4 | Pass |  |

Table 5: Testing the Acceptance Tests

### 6.5.2 AI Performance Testing on Real-World Data

In this section, the performance of the CNN will be tested to determine its effectiveness on real-world plants. Each class of plant was tested using ten real-world plants of each type. Based on the requirements as well as test validation results, the estimated minimum score value should be 9 with a margin of error of ± 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Plant Type being Tested | Score (out of 10) | Pass or fail | Comments |
| Aglaonema | 9/10 | Pass |  |
| Aloe vera | 7/10 | Fail | When this plant was misidentified, it was determined to be Dracaena Angolensis (a cylinder snake plant). This is most likely due to very similar leave shape (both consisting of long spear-like leaves) and canopy structure these plants share. |
| Calathea | 9/10 | Pass |  |
| Chlorophytum Comosum | 10/10 | Pass |  |
| Crassula Ovata | 6/10 | Fail | This was determined to be due to the dataset itself not including enough examples of adult jade plants (CrassulaOvata), as whilst undertaking further testing it was concluded that the AI model worked effectively with younger specimens with an accuracy of 9/10 on young plant specimens, but could rarely identify adult jade plants, only scoring 4/10 on a test set on only adult jade plant specimens. |
| Dracaena Angolensis | 10/10 | Pass |  |
| Dracaena Warneckii | 9/10 | Pass |  |
| Fittonia Albivenis | 10/10 | Pass |  |
| Janda Bolong | 9/10 | Pass |  |
| Keladi | 8/10 | Pass | Whilst lower than the ideal minimum, these results do fall in the margin of error. |
| Schefflera | 9/10 | Pass |  |

Table 6: Testing the Real-World Performance of the CNN

# 7. Project Evaluation

In this section, each phase of the project will be evaluated.

## 7.1 Evaluation of the Research Phase

### 7.1.1 What Went Well?

The literature review section of this project went well, with effective research undertaken with comparative studies used wherever possible to ensure that information was effectively evaluated ensuring that accurate information was used. The requirement gathering and analysis section of this project also went well due to the effective conclusion of the aims and objectives of the project alongside the extensive research undertaken. This made this section simple to undertake and allowed for all requirements of the system to be effectively justified by the aims and objectives, and the literature review. The software methodology section of the project also went well, with a primary focus on ensuring the effective balance between the implementation of software and report writing resulting in an effective hybrid methodology being chosen resulting in a superior outcome for the project.

### 7.1.2 Limitations

The main limitation was the lack of previous research specifically focused on houseplant identification, meaning that the assumption that information about general plant identification would also apply to this subtopic had to be made, which whilst still resulting in an overall effective literature review, did mean potential methods of identification being missed, resulting in a less robust project in comparison.

### 7.1.3 Improvements and Lessons Learnt

The main area of improvement to this phase would be to have an additional focus on research of application design for all major mobile platforms, rather than the specific focus made on Android development, this would have allowed for a greater knowledge of effective mobile interface development to be obtained.

The main lesson learnt from this phase of the project is that it is very important to not pigeon-hole oneself down a certain road of inquiry, as this results in arbitrary restrictions being placed on the software before the design or requirements have taken place, making the development process restrictive and preventing the utilisation of beneficial information from other relevant sources that are not directly related to the developer’s vision of the system.

## 7.2 Evaluation of the Design Phase

### 7.2.1 What Went Well?

The overall system designs went well, which is mainly attributed to extensive requirements and the aims and objectives being fleshed out effectively enough to ensure that all its functionality was known in detail in advance. The mobile application interface design also went well, thanks to the extensive and detailed list of requirements set out for the mobile application in advance. The database design also went well, again due to the extensive set requirements of the system making it clear what was expected of the database and how it would be achieved, making the process of designing the database straightforward. Finally, the assets design also went well and was arguably the most enjoyable part of the design phase, which assisted in making the design work for these assets more rewarding at the cost of a lot of potential assets design being left unused.

### 7.2.2 Limitations

The primary limitation of the design phase was the AI model design, despite the type of AI being known thanks to the research, the exact parameters and layers present in the CNN model could not be effectively concluded until the actual implementation as the model needed to adept and be calibrated to support its particular use case.

A secondary limitation was the lack of potential user feedback on designs, this would have been beneficial and would have potentially resulted in the overall design of the application being more user-orientated and I feel that the lack of feedback made the design process more insulated and tailored to the designers wants rather than any potential user.

### 7.2.3 Improvements and Lessons Learnt

The main lesson learnt is that extensive user feedback in the early stages of design and development is vital to ensure that the software being developed is effectively tailored to the target audience and assists in making up for oversights made by the software designer and/or software designing team as each audience looks at the software from very different points of view. To ensure these different views are aligned on the same goal of producing effective software, close collaboration between the two is needed.

## 7.3 Evaluation of the Implementation Phase

### 7.3.1 What Went Well?

The gathering of data for the custom-made datasets went well, with an extensive selection of plant specimens being obtained in a relatively brief time frame, whilst still being extensive enough and varied enough to ensure an overall high level of accuracy in the AI model.

The implementation of the Android application went well, especially the development of the user interface thanks to the extensive design work done prior, implementing it became simple.

The creation of the database was one of the smoothest parts of the implementation phase, with absolutely no issues being encountered whilst developing it, this can be attributed primarily to the simplistic nature of the database as well as the effective pre-planning in the design phase.

Finally, the integration of the two subsystems into the mobile application also went surprisingly well, I say this as the integration between different technology tends to be the hardest part of any project due to the prevalent difficulty of intercommunication between software, but thanks to extensive user documentation, this process ran smoothly.

### 7.3.2 Limitations

The biggest limitation of this phase was with hardware which made this phase more limited especially since the long implementation time of the CNN had to be balanced with the implementation of other parts of the system and other university work. This balance resulted in an overall compromise of the scope of the model’s implementation, with training sets having to be made smaller, overall making the AI model less robust.

Another limitation was the lack of pre-existing datasets. This resulted in a section of the project that was initially expected to be a short footnote in the overall training of the TensorFlow Lite model becoming more significant as the requirement for the implementation of custom-made houseplant datasets became needed. Whilst beneficial from the mindset of writing this report, this did result in the implementation running behind schedule almost immediately.

### 7.3.3 Improvements and Lessons Learnt

Despite the system meeting all specified requirements many improvements could be made, primarily this would be focused on useability, the overall functionality of the AI model and database, with the addition of more types of plants that the application could identify, as well as work on the compatibility of the application between different OS i.e., iOS.

The primary lesson to learn from this phase of the project is to always ensure every sensible potential implementation of a feature has been researched in-depth before implementation, especially when it comes to AI. This lesson has been learnt from the issues that occurred when attempting to convert the TensorFlow model into a TensorFlow Lite model, as, if the method of directly building the AI model as a TensorFlow Lite was discovered earlier, less time would have been spent on an inferior method.

This leads to the next lesson, that the specified documentation is not always the best method to solely follow, and that a combination of both the official documentation and the resources provided by those who use the software should be followed in tandem as it opens more points of information to explore and utilise.

## 7.4 Evaluation of the Testing Phase

### 7.4.1 What Went Well?

The system is built like a Rube Goldberg machine where one simple input sets off a complex interconnected chain reaction of functionality, making testing the application succinct as for any accurate output to be achieved each system must be working fully and effectively to achieve it.

### 7.4.2 Limitations

The main limitation of the testing phase was the assumptions needed to be made when it came to testing the AI model. This is due to the black box model nature of AI where the exact parameters and processes going on inside the AI model have very little way to be effectively visualised and it becomes difficult to diagnose how something has gone wrong, meaning assumptions must be made of why an error occurred making correcting errors with the AI model particularly difficult.

### 7.4.3 Improvements and Lessons Learnt

Whilst more extensive testing including but not limited to unit tests could have been undertaken, a key lesson learnt in this phase is that early development of tests alongside the requirements is highly beneficial as assuming the system has been designed effectively to meet those requirements the testing becomes a quick process making it easier to ensure that the system has met all the requirements.

# 8. Conclusion and Further Work

## 8.1 Conclusion

The best way to conclude the project is to start where we began, the aims and objectives.

These first three aims and objectives have been met with the final product of the mobile application consisting of an AI model and database.

This issue with the fourth objective is that the field of AI is rapidly changing with new methods being discovered and developed frequently, meaning that the time of writing this objective, as concluded from the literature review, has been met, however, this soon may be not the case as the relatively young age of the field of AI image classification means a new AI technique could soon supersede the CNN method.

With the literature review providing the basis of which AI optimisation algorithm to use and the comparative implementations in the implementation phase between the Adam and the Adamax optimisation algorithm, it can be concluded that the fifth aim and objective has been successfully met, with both the literature review and the comparative implementation supporting the findings that Adamax is the better optimisation algorithm in this use case.

The sixth objective is difficult to quantify because despite concluding from the literature review and the implementation of the system that an effective method to use is to directly create a TensorFlow Lite model, this was only concluded for platforms running the Android OS. This means that this may not be the best method for other mobile/IoT operating systems. So arguably this has been met, with the caveat that further work is required to conclude if this method also applies to other devices.

## 8.2 Further Work

In this section, future work and improvements to the project are discussed.

### 8.2.1 Expansion of the Database, the AI Model, & the Datasets

The main objective of this improvement would be to increase the number of houseplants the system can effectively identify, for each new plant the AI model can identify, a new entry in the plant care database would need to be created. This also brings up the idea of further expanding the size of the datasets used for each plant in the system, increasing them to at least 150 images, as this will assist in making the model more accurate.

### 8.2.2 Mobile Application Quality of Life Improvements

Due to the relatively simplistic nature of the mobile application interface, several quality-of-life improvements could be made, this includes but are not limited to, adding support for additional languages other than English, which would allow for greater use of the application worldwide and support users who do not natively speak English. Another improvement that will be covered is the ability to upload images into the application that are stored on the user’s device, this would allow for the application to still be effectively used on devices with broken or no camera functionality as well as making the application more flexible to the wants and needs of the user.

### 8.2.3 System Implementation on Other Platforms

As stated in the requirements as well as shown in the design an iOS version of the application was intended to be developed, this would be a native application to ensure effective integration between the AI model and iOS. The primary issue for this is that Apple is rather restrictive with the use of AI on their devices, due to their walled garden approach to application on their devices, requiring extensive testing to be undertaken, this time commitment, as well as an overall lack of experience with iOS development, was the primary reason this was not undertaken, however, this would be the ideal time to port the application over, of course, changes for compatibility will need to be made, for example, the use of the Swift programming language instead of Kotlin and an extensive UI recoding to meet Apple's design methods and standards.

# 9. References / Bibliography

* Aakif, A. and Khan, M.F. (2015) Automatic Classification of Plants Based on Their Leaves. Biosystems Engineering [online]. 139 (1), pp. 66-75. [Accessed 25 January 2022].
* Abdullah S. M., Halimi M., Karamizadeh S., Rajabi M. J., and Shayan J. (2014) Advantage and drawback of support vector machine functionality. In: (Unknow editor) ed. 2014 International Conference on Computer, Communications, and Control Technology (I4CT). Langkawi, Malaysia, 2-4 Sept. IEEE, pp. 63-65
* Akata, Z., Harchaoui, Z., Perronnin, F. and Schmid, C. (2014) Good Practice in Large-scale Learning For Image Classification. Transactions on Pattern Analysis and Machine Intelligence [online]. 36 (3), pp. 507-520. [Accessed 28 January 2022].
* Al-Murad A., Islam K. T., and Raj R. G. (2017) Performance of SVM, CNN, and ANN with BoW, HOG, and Image Pixels in Face Recognition. In : (Unknow editor) ed. 2017 2nd International Conference on Electrical Electronic Engineering (ICEEE). Rajshahi, Bangladesh, 27-29 Dec. IEEE, pp. 1-4
* Alsing, O. (2018) Mobile Object Detection using TensorFlow Lite and Transfer Learning, [online]. MSc, KTH Royal Institute of Technology. Available from: https://www.diva-portal.org/smash/get/diva2:1242627/FULLTEXT01.pdf [Accessed 30 January 2022].
* Aptoula, E., Ghazi, M.M. and Yanikoglu, B. (2017) Plant Identification Using Deep Neural Networks Via Optimization of Transfer learning Parameters. Neurocomputing [online]. 235 (1), pp. 228-235. [Accessed 23 October 2021].
* Arfin M. H. R., Hossain S. A., Islam M. M., and Rabby S. A. (2019) PataNET: A Convolutional Neural Networks to Identify Plant from Leaf Images, 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Daffodil International University, Kanpur, India, 6-8 July 2019, IEEE [online]. Available from: https://ieeexplore-ieee-org.ezproxy.uwe.ac.uk/document/8944667 [Accessed 24 October 2021].
* Arora, S. and Kour, V.P.(2019) Particle Swarm Optimization Based Support Vector Machine (P-svm) For the Segmentation and Classification of Plants. Ieee Access [online]. 7 (1), pp. 29374-29385. [Accessed 22 January 2022].
* Arshid, K., Feng, J., Jia, K., Mehmood, A., Rehman, Z.U., Yaqub, M. and Zia, M.S. (2020) State-of-the-art Cnn Optimizer For Brain Tumor Segmentation in Magnetic Resonance Images. Brain Sciences [online]. 10 (7), pp. 427-447. [Accessed 29 January 2022].‌
* Awang, K, Che Hussin, N.A., Jamil, N. and Nordin, S. (2013) Plant species identification by using Scale Invariant Feature Transform (SIFT) and Grid Based Colour Moment (GBCM). In: (Unknown), ed. 2013 IEEE Conference on Open Systems. Kuching, Malaysia, 2-4 Dec. IEEE, pp. 226-230
* Balasaravanan T., Priya C.A, and Thanamani A.S. (2012) An efficient leaf recognition algorithm for plant classification using support vector machine. In: (Unknown Editor), ed. International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME-2012). Salem, India, 21-23 March. IEEE, pp. 428-432
* Bringslimark, T., Hartig, T., and Patil, G.G., (2009) The Psychological Benefits of Indoor Plants: A Critical Review of the Experimental Literature. Journal of Environmental Psychology [online]. 29 (4), pp. 422-433. [Accessed 01 January 2022].
* Butler, C.W. and Vijayasarathy, L.R. (2016) Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter?. Ieee Software [online]. 33 (5), pp. 86-94. [Accessed 31 January 2022].
* Cham, W.K., Chan, S.S.F., Chu, L.M. and Zhao, C. (2015) Plant Identification Using Leaf Shapes—a Pattern Counting Approach. Pattern Recognition [online]. 48 (10), pp. 3203-3215. [Accessed 21 January 2022].
* Chan C. S., Lee H., Remagnino P., and Wilkin P. (2015) Deep-plant: Plant identification with convolutional neural networks. 2015 IEEE International Conference on Image Processing (ICIP), University of Malaya, Royal Botanic Gardens, & Kingston University, Quebec City, QC, Canada, 27-30 Sept. 2015: IEEE [online]. Available from: https://ieeexplore.ieee.org/abstract/document/7350839 [Accessed 23 October 2021].
* Chao, X., Li, Y. and Nie, J. (2020) Do We Really Need Deep Cnn For Plant Diseases Identification?. Computers and Electronics in Agriculture [online]. 178 (1), pp. 1-7. [Accessed 28 January 2022].
* Choo K.R., Huang F., Liu P. and Wang L. (2017) Svm Or Deep Learning? a Comparative Study on Remote Sensing Image Classification. Soft Computing [online]. 21 (1), pp. 7053-7065. [Accessed 25 January 2022].
* Gajjar, R., Gajjar, N., Patel, N.P., Thakor, V.J. and Ruparelia, S. (2021) Real-time Detection and Identification of Plant Leaf Diseases Using Convolutional Neural Networks on an Embedded Platform. The Visual Computer [online]. 1 (1), pp. 1-16. [Accessed 26 October 2021].
* Garleanu, A., Mărzan, M., Paul, C., Spiru, L., and Velciu, M. (2019) THE REVERSED MOSCOW METHOD. A GENERAL FRAMEWORK FOR DEVELOPING AGE-FRIENDLY TECHNOLOGIES. In: (editor Unknown) ed. 11th International Conference e-Health 2019. Porto, Portugal, 17 – 19 July. Inderscience Publishers, pp. 75- 81.
* Jones, D.D., Meyer, G.E., Neto, J.C. and Samal, A.K. (2006) Plant Species Identification Using Elliptic Fourier Leaf Shape Analysis. Computers and Electronics in Agriculture [online]. 50 (2), pp. 121-134. [Accessed 21 January 2022].
* Karthikeyan, N.G (2018). Machine Learning Projects for Mobile Applications: Build Android and IOS Applications Using TensorFlow Lite and Core ML [online]. Birmingham: Packt Publishing Ltd. [Accessed 30 January 2022]
* Lalithnarayan, C. (2020) Handling Imbalanced Datasets in Machine Learning. Available from: https://www.section.io/engineering-education/imbalanced-data-in-ml/ [Accessed 13 February 2022].
* Lu, W.Z. and Wang, W.J. (2005) Potential Assessment of the “Support Vector Machine” Method in Forecasting Ambient Air Pollutant Trends. Chemosphere [online]. 59 (5), pp. 693-701. [Accessed 25 January 2022].
* Macario V., Oliveira J. F. L., and Pacifico L. D. S. (2018) Plant Classification Using Artificial Neural Networks. In: (unknown editor) ed. 2018 International Joint Conference on Neural Networks (IJCNN). Rio de Janeiro, Brazil, 8-13 July. IEEE, pp. 1-6
* Okamoto, K., Tanno, R. and Yanai, K. (2016) Efficient Mobile Implementation of A CNN-based Object Recognition System. In: (Unknow editor) ed. 24th ACM international conference on Multimedia. Amsterdam, The Netherlands, October 15 – 19. Association for Computing Machinery, pp. 362–366
* Patle, A. and Prajapati, G. L. (2010) On Performing Classification Using SVM with Radial Basis and Polynomial Kernel Functions In: (Unknow editor) ed. 2010 3rd International Conference on Emerging Trends in Engineering and Technology. Goa, India, 19-21 Nov. IEEE, pp. 512-515
* Rao, T. V. M. and Vani, S. (2019) An Experimental Approach towards the Performance Assessment of Various Optimizers on Convolutional Neural Network In: (Unknow editor) ed. 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI). Tirunelveli, India, 23-25 April. IEEE, pp. 331-336.
* Royal Botanic Gardens (2020) State of the World’s Plants and Fungi 2020 [online]. London, UK: Royal Botanic Gardens, Kew. [Accessed 21 January 2022].
* Zhang, Z. (2018) Improved Adam Optimizer for Deep Neural Networks. In: (unknown editor) ed. 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS). Banff, AB, Canada, 4-6 June. IEEE, pp. 1-2.

# 10 Appendix

## 10.1 Requirement Analysis

### 10.1.1 Additional User stories

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | As a… | I want… | So that… | MoSCoW |
| US.AI6 | User | To have my plant identified using a picture. | I can easily give the information needed for my plant to be identified | M |
| US.AI7 | User | To be able to have my plants identified with a portable device | I can identify my plant on an easily accessible device as well as on the go | M |
| US.AI8 | User | To be able to have the number of different houseplants identifiable easily expanded upon | Over time as more plants become available, I can continue to use the system. | S |
| US.AI9 | User | The identification method to be easily explainable | I can understand how my plant is being identified | C |
| US.UI1 | User | To be able to upload images of my plant to an application for identification | I can have means of quick and portable plant identification | M |
| US.UI2 | User | To be able to upload pictures of my unknown plant from files stored on my device | I can identify plants I’ve found online or overwise do not have direct access to | S |
| US.UI3 | User | To be able to take pictures of my unknown plant, and upload them, through the system | I can identify any plant I have direct access to | M |
| US.UI4 | User | To have the uploaded image of my unknown plant displayed in the system | I can see my plant to allow me to associate more effectively what it is with an example of the plant | M |
| US.UI5 | User | To have information displayed about how to care for my identified plant | I can effectively take care of my newly identified plant | M |
| US.UI6 | User | Plant information to be displayed in the form of a table | I can see information about my plant in a concise format | M |
| US.UI7 | User | To be able to upload multiple consecutive images of different plants | I can identify multiple different plants without restarting the system | M |
| US.UI8 | User | To be able to queue up photos of different plants to have them identified | I can identify multiple different plants all at the same time. | C |

Table 7: The Additional User Stories

### 10.1.2 Additional Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirements | Priority (MoSCow) | Source/ Justification |
| AI6 | The AI system must be able to handle poor quality images of houseplants | S | Core functionality |
| AI7 | The AI system implemented must consist of a CNN | M | Literature review: Arfin et al (2019), Aptoula, Ghazi, and Yanikoglu (2017), Gajjar et al (2021), and Chan et al (2015) |
| AI8 | Two CNNs must be created, one using ADAM optimisation, the other using Adamax optimisation, to determine the best method | M | Literature review: Rao and Vani (2019), and Arshid et al (2020) |
| AI9 | The CNN will be able to support full colour images | S | Literature review: Awang *et al*. (2013) |
| AI10 | The CNN will be able to be portable and lightweight enough to function on a mobile platform | M | Core functionality and Literature review: Alsing (2018) and (Karthikeyan, 2018) |
| UI6 | The user interface must allow the user to do repeat submissions of different images to the CNN | M | Core functionality |
| UI7 | The user interface must display the correct image of the given plant identified by the CNN | M | Core functionality |
| UI8 | The user interface must display the name of the identified plant | M | Core functionality |
| UI9 | The user interface will need to be able to clear old identification requests | M | Core functionality |
| UI10 | The user interface must scale input images down to 180x180 before passing it to the CNN | S | Literature review: Okamoto, Tanno, and Yanai (2016) |
| DB6 | The database**'**s plant information table will need a column to store the amount of water a plant needs. | M | Core functionality |
| DB7 | The databases plant information table will need a column to store the ideal temperature for a plant | M | Core functionality |
| DB8 | The databases plant information table will need a column to store the amount of potting space needed | S | Core functionality |
| DB9 | The databases plant information table will need a column to store the soil type needed | S | Core functionality |
| DB10 | The databases plant information table will need a column to store any plant nutritional requirements | S | Core functionality |

Table 8: The Additional Functional Requirements

### 10.1.3 Addition Non-Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirements | Priority (MoSCow) | Source/ Justification |
| NF.AI6 | The CNN must be implemented through TensorFlow Lite, to allow for effective mobile integration | M | Core functionality & Literature review: Alsing (2018) & Karthikeyan (2018) |
| NF.UI1 | All interactions with the user interface must respond within 1 second of interaction | M | Core functionality |
| NF.UI2 | All information displayed through the app must be in English | M | Core functionality |
| NF.UI3 | All information displayed through the user interface must be clear and readable | M | Core functionality |
| NF.UI4 | All interaction with the user interface must be clear and understandable | M | Core functionality |
| NF.DB1 | Once the plant is identified, the database must respond and return the relevant care information within 2 seconds | M | Core functionality |
| NF.DB2 | Any data modification made to the database must be updated to all users within 3 seconds of the update occurring | M | Core functionality |
| NF.DB3 | The database must be easily maintainable, allowing for new information to be easily added, altered, or deleted | M | Core functionality |
| NF.DB4 | All information delivered from the database will consist of standard English, all spelt correctly | M | Core functionality |
| NF.DB5 | The database must be implemented with SQLite, to allow for mobile integration | M | Core functionality |
| NF.All1 | From initial submission of the plant image, the system must return both the plants identity (if it can) and the relevant information from the database within 3 seconds | M | Core functionality |
| NF.All2 | All services, including the CNN, mobile application and the database must have an uptime of over 99% | M | Core functionality |
| NF.All3 | All software must be runnable on android devices | M | Core functionality |
| NF.All4 | The application must be able to run on at least 50% of android devices currently on the market | S | Core functionality |
| NF.All5 | All software must be runnable on iOS devices | C | Core functionality |
| NF.All6 | The application must be able to run on at least 50% of iOS mobile devices on the market | C | Core functionality |

Table 9: The Additional Non-Functional Requirements