

**APPLICATION DEVELOPMENT FOR
TRADITIONAL CHINESE MEDICINE HERB**

By
Loke Zhu Mun

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

FEBRUARY 2025

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr Tan Joi San who has given me this bright opportunity to engage in an IC design project. It is my first step to establish a career in IC design field. A million thanks to you.

To a very special person in my life, Wong Chi Heang, for her patience, unconditional support, and love, and for standing by my side during hard times. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

COPYRIGHT STATEMENT

© 2025 Loke Zhu Mun. All rights reserved.

This Final Year Project proposal is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project proposal represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project proposal may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ABSTRACT

Traditional Chinese Medicine (TCM) has been practiced for thousands of years, relying heavily on the accurate identification and combination of herbs for effective treatment. However, the vast array of herbs and their subtle differences pose significant challenges in ensuring accurate identification and proper use, especially for non-experts. To address these challenges, this project proposes the development of a mobile application that utilizes advanced image recognition technology, specifically Convolutional Neural Networks (CNNs), to accurately identify TCM herbs and provide users with personalized herb combination recommendations.

The application is designed to enhance the accessibility and safety of TCM practices by offering a user-friendly interface that supports herb identification, detailed information on each herb, and symptom-based herb recommendations. The project leverages the capabilities of a pre-trained CNN model, fine-tuned with a comprehensive herb dataset, to deliver high accuracy in herb recognition. Additionally, the application includes features such as user authentication, camera integration for herb scanning, and a history tracking module to support personalized user experiences.

Through this project, the application aims to modernize TCM practices by integrating cutting-edge technology, providing a valuable tool for practitioners, students, and patients to enhance their understanding and effective use of TCM herbs.

Area of Study: Application Development, Deep Learning

Keywords: Mobile Application, Deep Learning, TCM Herb recognition, Mobile Scanner, Symptom-Based Recommendation

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	ii
COPYRIGHT STATEMENT	iii
ABSTRACT	iv
LIST OF FIGURES	viii
LIST OF TABLES	xii
LIST OF ABBREVIATIONS	xiii
CHAPTER 1 - Introduction	1
1.1 Problem Statement and Motivation	2
1.2 Research Objectives	3
1.3 Project Scope	3
1.4 Proposed Approach / Study	6
1.5 Highlight of what have been achieved	7
1.6 Report Organization	8
1.7 Summary	8
CHAPTER 2 - Literature Review	9
2.1 Previous Works on recognition of traditional Chinese medicine	9
2.1.1 Deep learning-enabled mobile application for efficient and robust herb image recognition	9
2.1.2 CCNNet: a novel lightweight convolutional neural network and its application in traditional Chinese medicine recognition	13
2.2 Limitations of Previous Studies	15
2.3 Existing Mobile Applications	16
2.3.1 Herbapp	16
2.3.2 Traditional Chinese Medicine	21
2.3.3 TCM Clinic Aid Trial	24
2.4 Summary	27
CHAPTER 3 – System Design	29
3.1 Use-Cases Diagram	29
3.2 Use-case Description	30

3.3	Program Development	38
3.3.1	Login Page Development	38
3.3.2	Register Page Development	40
3.3.3	Herb type Page Development	42
3.3.4	Formula type Page Development	44
3.3.5	Herb Detail Page Development	46
3.3.6	Formula Detail Page Development	48
3.3.7	Notes Page Development	50
3.3.8	Record Page	52
3.3.9	Profile Page Development	53
3.3.10	Herb Classification Page Development	55
3.3.10.1	Hugging Face API for Single Scan	60
3.3.10.2	Hugging Face API for Multi Scan	61
3.3.11	Model Architecture	63
3.3.11.1	Efficient Net Model	63
3.3.11.2	YOLO Architecture	64
3.4	Summary	65
CHAPTER 4	– Methodology and Tools	66
4.1	Methodologies	66
4.1.1	Requirements Planning Phase	66
4.1.2	User Design	67
4.1.3	Construction and Feedback Phase	67
4.1.4	Finalize Product and Implementation Phase	68
4.2	Tools to Use	69
4.2.1	Hardware	69
4.2.2	Software	70
4.3	Gantt Chart	72
4.4	Summary	73
CHAPTER 5	– Implementation and Testing	74
5.1	Login Page	74
5.2	Register Page	75
5.3	Herb Type Page	76

5.4	Herb Detail Page	78
5.5	Formula Type Page	79
5.6	Formula Detail Page	80
5.7	Record Page	81
5.8	Notes Page	83
5.9	Profile Page	84
5.10	Herb Classification Page	85
5.11	TensorFlow Model	90
5.11.1	Dataset	90
5.11.2	Splitting the Dataset	90
5.11.3	Data Augmentation	91
5.11.4	Model Architecture	93
5.11.5	Model Training	93
5.11.6	Performance Evaluation	95
5.11.7	Result	99
5.11.8	FYP 1 vs FYP 2 Comparison	102
5.12	YOLO Model	102
5.12.1	Dataset	103
5.12.2	Splitting the dataset	104
5.12.3	Data Augmentation	104
5.12.4	Model Training	106
5.12.5	Performance Evaluation	106
5.12.6	Result	110
5.13	Conclusion	112
CHAPTER 6 – CONCLUSION		113
6.1	Project Review, Discussion and Conclusion	113
6.2	Novelties and Contributions	114
6.3	Future Work	114
REFERENCES		116

LIST OF FIGURES

Figure 1.4. 1: Flowchart of Herb Scanner App.	6
Figure 2.1.1. 1 : Visualization of data augmentation techniques [3].	10
Figure 2.1.1. 2: Overview of the network compression approach [3].	11
Figure 2.1.2. 1: Architecture of CCNNNet [4].	14
Figure 2.3.1. 1: Main Page of Hearbapp.	17
Figure 2.3.1. 2: Page of Herb.	18
Figure 2.3.1. 3: Page of interactions in herb.	18
Figure 2.3.1. 4: Page of related formulas of the herb.	19
Figure 2.3.1. 5: Page of formula.	19
Figure 2.3.1. 6: Page of herbs in formula.	20
Figure 2.3.1. 7: Page of flash card decks.	20
Figure 2.3.2. 1: Main page of Traditional Chinese Medicine.	22
Figure 2.3.2. 2: Information about the types of drugs.	22
Figure 2.3.2. 3: Information of the herb.	23
Figure 2.3.3. 1: Main page of TCM Clinic Aid Trial.	24
Figure 2.3.3. 2: Search by Symptoms.	25
Figure 2.3.3. 3: Search by materia medica.	25
Figure 2.3.3. 4: Page of Individual Herbs.	26
Figure 2.3.3. 5: Information of herbs.	26
Figure 3.1. 1: Use-cases diagram.	29
Figure 3.3.1. 1: Code of email sign in function.	39
Figure 3.3.1. 2: Code of google sign in function.	39
Figure 3.3.1. 3: Code of reset password function.	40
Figure 3.3.1. 4: Code of save login status.	40
Figure 3.3.2. 1: Code of register function.	41
Figure 3.3.2. 2: Code of register function 2.	41
Figure 3.3.3. 1: Load herb data from Herb JSON file function.	42
Figure 3.3.3. 2: Herb JSON file.	43
Figure 3.3.3. 3: Code of herb search bar.	43
Figure 3.3.3. 4: Code of herb filter by properties.	44

Figure 3.3.4. 1:Code of Category filter.	45
Figure 3.3.4. 2: Formula JSON file.	45
Figure 3.3.4. 3: Code of formula filter by symptoms.	46
Figure 3.3.5. 1: Part of display herb detail section.	47
Figure 3.3.5. 2: Function Card to display herb data.	47
Figure 3.3.6. 1: Part of display formula detail.	48
Figure 3.3.6. 2: Code of herbs card.	49
Figure 3.3.6. 3: A part code navigates to herb detail.	49
Figure 3.3.7. 1: Code of note list page.	51
Figure 3.3.7. 2: Code of note editor page.	51
Figure 3.3.8. 1: Code of load both storage classes.	52
Figure 3.3.8. 2: Code of toggling logic.	53
Figure 3.3.9. 1: Load user data function.	54
Figure 3.3.9. 2: App version function.	54
Figure 3.3.9. 3: Log out function.	54
Figure 3.3.10. 1: Code of change to mode.	55
Figure 3.3.10. 2: Code of crop center.	55
Figure 3.3.10. 3: Code to process API to recognise.	56
Figure 3.3.10. 4: Code to link the hugging face API.	56
Figure 3.3.10. 5: Code of show single mode result.	57
Figure 3.3.10. 6: Code to match the herb and the formula.	57
Figure 3.3.10. 7: Code of navigate to formula details page.	58
Figure 3.3.10. 8: Code to handle no formula matched.	59
Figure 3.3.10. 9: Code to load the 2 JSON data.	60
Figure 3.3.10.1. 1: Single scan hugging face.	61
Figure 3.3.10.1. 2: Process of Efficient Net predict images.	61
Figure 3.3.10.2. 1: Multi scan hugging face.	62
Figure 3.3.10.2. 2: Process of YOLO predict images.	62
Figure 3.3.11.1. 1: EfficientNet-B3 Architecture.	63
Figure 3.3.11.2. 1: YOLO Architecture.	64
Figure 4.1. 1: Process of RAD [5].	66
Figure 4.3. 1: Gantt Chart 1.	72

Figure 4.3. 2: Gantt Chart 2.	73
Figure 5.1. 1: Login Page.	75
Figure 5.2. 1: Register Page.	76
Figure 5.3. 1: Herb Type Page.	77
Figure 5.3. 2: Filter of herb's properties.	77
Figure 5.4. 1: Herb Detail Page.	78
Figure 5.5. 1: Formula type page.	79
Figure 5.5. 2: Filter of symptoms.	80
Figure 5.6. 1: Formula detail page.	81
Figure 5.7. 1: Record page in herbs.	82
Figure 5.7. 2: Record page in formulas.	82
Figure 5.8. 1: Notes page 1.	83
Figure 5.8. 2: Notes page 2.	84
Figure 5.9. 1: Profile Page.	85
Figure 5.10. 1: Single scan.	86
Figure 5.10. 2: Single scan successfully.	87
Figure 5.10. 3: Multi scan.	88
Figure 5.10. 4: Multi scan successfully.	88
Figure 5.10. 5: Multi scan no match formula.	89
Figure 5.11.1. 1: Total dataset	90
Figure 5.11.2. 1: Split data to 70% training set and 30% testing set.	91
Figure 5.11.2. 2: Distribution of dataset after splitting.	91
Figure 5.11.3. 1: Augmentation and resize code.	92
Figure 5.11.3. 2: Sample of augmentation and resize.	92
Figure 5.11.4. 1: EfficientnetB3 Model Architecture.	93
Figure 5.11.5. 1: Build model function.	94
Figure 5.11.5. 2: Model training parameters.	94
Figure 5.11.6. 1: Training and validation accuracy.	95
Figure 5.11.6. 2: Training and validation loss.	95
Figure 5.11.6. 3: ROC Curve.	96
Figure 5.11.6. 4: Precision-Recall Curve.	97
Figure 5.11.6. 5: Confusion Matrix.	98

Figure 5.11.6. 6: Classification Report.	99
Figure 5.11.7. 1: Random result in test set from same class.	100
Figure 5.11.7. 2: Input manual and predict.	101
Figure 5.12.1. 1: Dataset of YOLO model.	103
Figure 5.12.2. 1: Splitting of the dataset.	104
Figure 5.12.3. 1: Resize of the dataset.	105
Figure 5.12.3. 2: Augmentation of the dataset.	105
Figure 5.12.4. 1: YOLO training.	106
Figure 5.12.5. 1: YOLO F1-Confidence Curve.	107
Figure 5.12.5. 2: YOLO Precision-Recall Curve.	107
Figure 5.12.5. 3: YOLO Precision-Confidence Curve.	108
Figure 5.12.5. 4: YOLO Recall-Confidence Curve.	108
Figure 5.12.5. 5: YOLO Confusion Matrix.	109
Figure 5.12.5. 6: Performance of the YOLO.	109
Figure 5.12.6. 1: Result of YOLO 1.	110
Figure 5.12.6. 2: Result of YOLO 2.	110
Figure 5.12.6. 3: Result of YOLO 3.	111
Figure 5.12.6. 4: Result of YOLO 4.	111

LIST OF TABLES

Table 2.1.1. 1: Performance evaluation of three smartphones [3].	12
Table 2.1.2. 1: Performance comparison of baseline models [4].	14
Table 2.1.2. 2: Evaluation of CCNNet expanded models across four datasets [4].	15
Table 2.4. 1: Summary of strengths and weaknesses.	28
Table 3.2. 1: Login use case description.	30
Table 3.2. 2: Register use case description.	30
Table 3.2. 3: Reset passwords use case description.	31
Table 3.2. 4: View type of herb use case description.	32
Table 3.2. 5: View detail of herb use case description	32
Table 3.2. 6: View type of formula use case description.	33
Table 3.2. 7: View detail of formula use case description.	33
Table 3.2. 8: Scan herb use case description.	34
Table 3.2. 9: View the record use case descriptions.	35
Table 3.2. 10: Write notes use case descriptions.	36
Table 3.2. 11: View profile use case description	37
Table 4.2.1. 1: Specifications of laptop.	69
Table 4.2.1. 2: Specifications of mobile device.	69
Table 4.2.2. 1: Specifications of software.	70
Table 5.11.8. 1: Table of comparison.	102

LIST OF ABBREVIATIONS

<i>AP</i>	Average Precision
<i>CNN</i>	Convolutional Neural Network
<i>CNNs</i>	Convolutional Neural Networks
<i>DNN</i>	Deep Neural Network
<i>FPS</i>	frames per second
<i>FYP</i>	Final Year Project
<i>GCIR</i>	Group Convolution Inverted Residual
<i>MDCA</i>	Multi-Dimension Channel Attention
<i>NAS</i>	Neural Architecture Search
<i>RAD</i>	Rapid Application Development
<i>ROC</i>	Receiver Operating Characteristic
<i>TCM</i>	Traditional Chinese Medicine
<i>YOLO</i>	You Only Look Once

CHAPTER 1 - Introduction

Traditional Chinese Medicine (TCM) is a medical system used to diagnose, treat, and prevent illnesses for more than 2,000 years. It is based on the philosophy of balancing the body's vital energy (Qi), Ying-Yang, and the Five Elements (Wood, Fire, Earth, Metal, Water) [1]. TCM encompasses a holistic approach to health and wellness, utilizing natural herbs, acupuncture, dietary therapy, and other practices to prevent and treat various ailments. Among these, one of the core components. These herbs are often combined in precise formulas to enhance their therapeutic effects and mitigate potential side effects [2].

Despite its long history and widespread use, TCM faces several challenges in the modern world. One significant challenge is the accurate identification of medicinal herbs. There are thousands of herbs used in TCM, each with distinct characteristics and therapeutic properties. Correct identification is crucial, as the efficacy and safety of TCM treatments largely depend on the precise selection and combination of these herbs.

In recent years, the rapid development of image recognition and machine learning technologies has opened new possibilities for addressing this challenge. Convolutional Neural Networks (CNNs), a class of deep learning models, have shown success in various image recognition tasks, including object detection, facial recognition, and medical imaging. CNNs are particularly well-suited for analyzing visual data due to their ability to automatically learn hierarchical feature representations from raw images.

Integrating CNN-based image recognition with TCM herb identification can significantly enhance the accessibility and accuracy of TCM practices. By leveraging mobile technology, we can develop a user-friendly application that allows users to simply take a picture of an herb and receive instant identification and combination recommendations. This can greatly benefit TCM practitioners, students, and patients, providing them with a reliable tool to support their practice and enhance their understanding of TCM.

1.1 Problem Statement and Motivation

Identifying and understanding TCM herbs can be challenging for those who are not experts. The lack of knowledge can lead to misidentification, incorrect usage, and potentially adverse health effects. Additionally, the absence of a user-friendly tool for accurate herb identification and guidance on their combinations and preparation methods further exacerbates this issue.

Given the importance of ensuring the safe and effective use of TCM herbs, there is a pressing need for reliable identification methods that are accessible to a broader audience. This challenge motivates the development of a solution that combines modern technology with traditional TCM knowledge, addressing the knowledge gap and promoting safe practices in TCM.

i. Identification of TCM Herbs

Many individuals, including practitioners and patients, struggle with both accurately identifying TCM herbs and understanding which herbs can be safely and effectively combined. The wide variety of herbs and their subtle differences make identification challenging, while the lack of reliable tools and guidance can lead to misidentification, misuse, and ineffective or potentially harmful combinations.

ii. Lack of Symptom-Based Herb Recommendation Tools

Beyond identification and combination challenges, there is also a significant gap in tools that allow users to input their symptoms or conditions to receive personalized TCM herb recommendations. Without such a system, users are left without adequate guidance on which herbs to use and how to combine them effectively based on their specific health conditions, potentially leading to suboptimal therapeutic outcomes or adverse effects.

1.2 Research Objectives

Develop a comprehensive mobile application that utilizes advanced image recognition technology to identify traditional Chinese medicinal (TCM) herbs and provide users with reliable herb combination recommendations. This application aims to enhance the accessibility and accuracy of TCM herb usage for both experts and non-experts.

i. Develop a TCM Herb Identification System

This project aims to create a comprehensive system that utilizes a fine-tuned Convolutional Neural Network (CNN) model to identify TCM herbs from user-uploaded images. The system will also recommend safe and effective herb combinations based on the identified herbs, incorporating a robust database that stores detailed information on herbs and their combinations. By integrating these features into a mobile application, the system will enhance users' understanding of effective TCM practices, improving their ability to safely use and combine herbs.

ii. Propose a Symptom-Based TCM Herb Recommendation System

This project will develop a system that allows users to input their symptoms or health conditions and receive tailored recommendations for suitable TCM herbs. The system will provide detailed guidelines on how to prepare and use these herbs, ensuring safe and effective treatment. By focusing on the user's specific health needs, this system aims to enhance the personalization and therapeutic effectiveness of TCM practices.

1.3 Project Scope

Develop a comprehensive mobile application that utilizes advanced image recognition technology to identify traditional Chinese medicinal (TCM) herbs. The application aims to provide accurate herb identification, detailed information about each herb, and personalized recommendations for herb combinations. By integrating a pre-trained Convolutional Neural Network (CNN) model and a user-friendly interface, the application will enhance accessibility to TCM and support users in making informed decisions about herb usage and combinations. After reading the literature, they have

some limitations, so this project aims to address the identified limitations by focusing on the trade-off between model size and accuracy. The project will refine existing models with enhanced attention mechanisms or employing transfer learning from large-scale pre-trained models could improve the models' ability to handle complex classifications while maintaining efficiency for mobile deployment.

i. Integration of Pre-trained Image Recognition Model

The application will utilize a pre-trained Convolutional Neural Network (CNN) model specifically suited for TCM herb identification. The pre-trained model will be fine-tuned with the pre-processed dataset to achieve high accuracy in recognizing different TCM herbs. This approach leverages existing advancements in image recognition technology to efficiently identify herbs in real-time within the mobile application.

ii. User Authentication and Profile Management Module

A secure and reliable user authentication system will be implemented to manage user access and protect personal data. Users will be able to create and manage their profiles, which will include personal preferences, saved herbs, and activity logs. This module ensures that each user has a personalized experience and can easily access their history and preferences.

iii. Camera Integration Module

The application will feature a camera interface that enables users to capture high-quality images of TCM herbs. This camera functionality will be designed to be user-friendly and optimized for capturing images of herbs, including features such as image stabilization and automatic focus.

iv. Herb Identification and Information Module

The fine-tuned CNN model will be integrated into the mobile application to enable herb identification. Upon identifying herbs, the application will provide users with detailed information, including the herb's name, properties, therapeutic effects, and usage guidelines. This module aims to offer users accurate and comprehensive information about the herbs they are interested in.

v. Symptom-Based Herb Recommendation Module

This module will be developed to allow users to input their symptoms or health conditions and receive tailored recommendations for suitable TCM herbs. The module will provide detailed explanations and guidelines for each recommended herb, including potential health benefits and preparation methods. This feature is designed to help users make informed decisions about using TCM herbs to address specific health issues effectively and safely.

vi. Record Module

The Record Module allows users to bookmark herbs manually while browsing the herb list. Each herb card includes a save icon, which users can tap to add the herb to their personal favourites list. This module helps users keep track of frequently used or personally relevant herbs, making it easier to revisit them later without needing to search again. The saved herbs are stored locally and accessible via the profile or favourites section.

vii. Notes Module

The Notes Module is an independent page that allows users to freely write and save general health notes or herbal observations. It functions like a digital notebook within the app, where users can jot down symptoms, treatment progress, or herbal preparation tips. The notes are stored locally and can be updated or edited at any time, providing users with a convenient way to document their herbal journey and wellness tracking.

1.4 Proposed Approach / Study

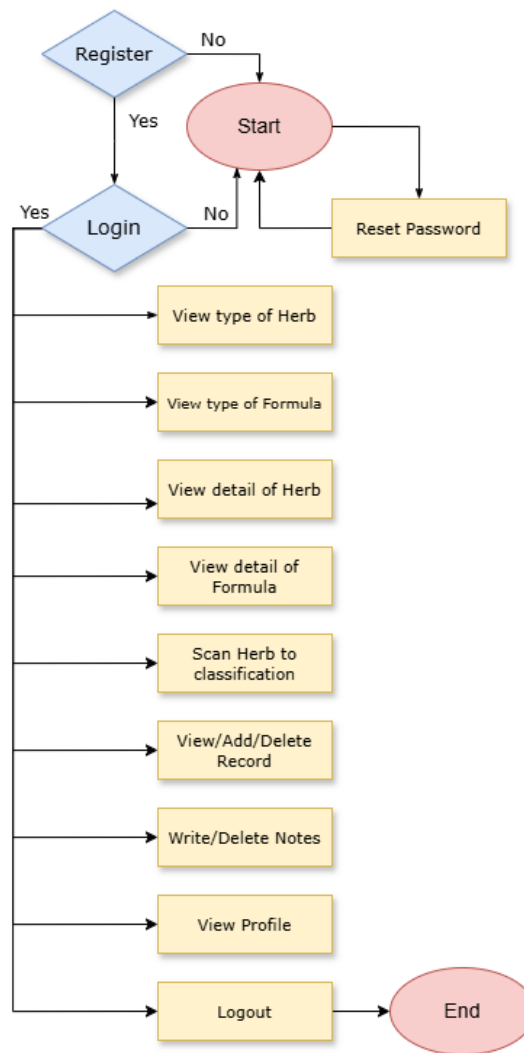


Figure 1.4. 1: Flowchart of Herb Scanner App.

The user interaction process of the Herb Scanner App is illustrated in Figure 1.4.1. The flowchart begins with three initial options: Register, Login, or Reset Password. These actions determine how users can access the main features of the app. New users must first complete the registration process, while existing users are required to log in. If a user forgets their password, the Reset Password function allows them to update their credentials by verifying their email.

Successful login is mandatory to access the application's core modules. If login fails, the system will prompt the user to re-enter their credentials. Once authenticated, users can proceed to explore the herb and formula categories, view detailed information for

each, and utilize the camera feature to perform herb classification via image input. Additional features include managing saved records, writing or deleting notes, viewing profile details, and securely logging out. This structured flow ensures a secure and personalized experience, allowing users to navigate all features only after successful authentication.

1.5 Highlight of what have been achieved

The Traditional Chinese Medicine (TCM) Herb Recognition Mobile Application was developed to provide a practical, AI-powered solution for herb identification and learning. The application successfully integrates a single-herb recognition feature using the EfficientNetB3 model and a multi-herb detection capability using YOLOv8, enabling users to classify individual or multiple herbs in real time with high confidence. The EfficientNetB3-based classifier achieved over 90% accuracy and an AUC of 0.99, while the YOLOv8 model obtained a mAP@0.5 of 0.94, validating its effectiveness in herb detection.

All planned modules for the application were successfully implemented. These include the User Authentication and Profile Management module for user login and personalization, the Camera Integration module for herb image capturing, and the Herb Identification and Information module that provides detailed explanations of each recognized herb. The Symptom-Based Herb Recommendation module allows users to input symptoms and receive matching formula suggestions. The Record module lets users bookmark herbs or formulas, and the Notes module enables users to document health notes or herbal insights directly within the app.

In addition, a curated dataset of 13 herb classes was built, annotated, and used for both classification and detection model training. Augmentation techniques and data preprocessing strategies were applied to optimize model performance. Overall, the project delivered a complete, intelligent mobile application that makes TCM herb learning accessible to the general public.

1.6 Report Organization

The details of this research are presented in the following chapters.

Chapter 2: Literature Review

Some related backgrounds and literature are reviewed to provide context for the study.

Chapter 3: Proposed method/Approach

It is discussing the hardware and software used, the methodology employed, system design components such as use case diagrams and program development, as well as implementation issues and challenges faced during the project. Additionally, the Gantt chart for project timeline management is included in this chapter.

Chapter 4: Preliminary Work

The preliminary work is detailed, highlighting how each page of the application is utilized and its functionalities.

Chapter 5: Conclusion

It concludes the report, summarizing the outcomes of the project and outlining plans for future development.

1.7 Summary

This project aims to modernize Traditional Chinese Medicine (TCM) practices by developing a mobile application that leverages a fine-tuned Convolutional Neural Network (CNN) for precise herb identification and personalized combination recommendations. Addressing the challenges posed by the wide variety and subtle differences among TCM herbs, the application improves accessibility, safety, and usability for practitioners, students, and patients. Key features include user authentication, camera integration for image-based herb recognition, symptom-based herb recommendations, record and notes, making TCM practices more efficient and user-friendly.

CHAPTER 2 - Literature Review

2.1 Previous Works on recognition of traditional Chinese medicine

2.1.1 Deep learning-enabled mobile application for efficient and robust herb image recognition

Xin Sun et. al. [3] discusses the development of a mobile application that utilizes deep learning for efficient and robust herb image recognition. The motivation for this study stems from the increasing popularity and necessity for quality control in herbal medicine. Traditional methods of herb recognition, which rely on expert knowledge and chemical processes, are inefficient, time-consuming, and impractical for resource-limited settings. This study proposed a solution by developing a lightweight deep learning model that can run on common smartphones, making herb recognition more accessible and feasible in such environments.

Deep learning was chosen for this application due to its success in image recognition tasks, particularly in cases where traditional methods are insufficient. Deep learning models, especially convolutional neural networks (CNNs), have demonstrated the ability to automatically extract features from images and classify them with high accuracy. This capability makes them suitable for handling the complex visual variations found in herbs. The challenge addressed by this paper is adapting these models to run efficiently on smartphones, which have limited computational resources compared to high-end servers.

The dataset used in this study is the largest publicly available herb image dataset, containing 95 categories with a total of 5640 images. These images were collected from the internet and feature mutual occlusion and cluttered backgrounds, making the classification task more challenging. The dataset was randomly split into 70% training and 30% testing sets in Figure 2.1.1.1, and fivefold cross-validation was used to evaluate the model's performance.

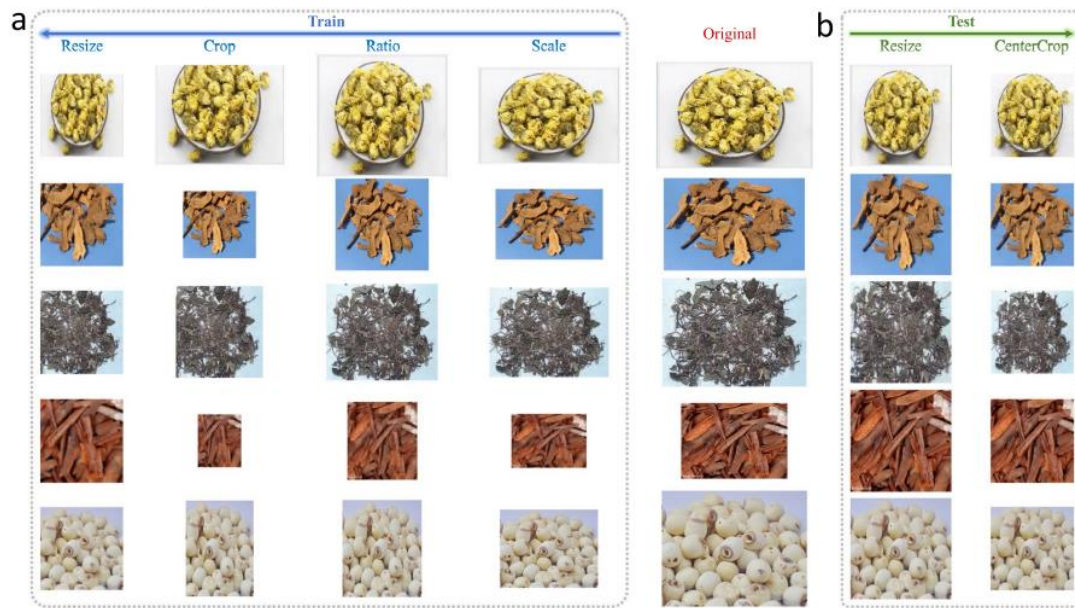


Figure 2.1.1. 1 : Visualization of data augmentation techniques [3].

The paper introduces a network compression algorithm in Figure 2.1.1.2 to reduce the size and computational requirements of deep learning models. The study starts with a standard deep learning model and compresses it into a smaller, more efficient version suitable for mobile deployment. The process involves three main steps: network pre-training on a large-scale dataset, network transfer to adapt the model to the specific task of herb recognition, and network cutting to further reduce the model size while maintaining accuracy.

Specifically, in Figure 2.1.1.2 (a), both large and small Deep Neural Networks (DNNs) are pre-trained on the ImageNet dataset and fine-tuned on the herb dataset; in Figure 2.1.1.2 (b), the large DNN's recognition capability is then transferred to the smaller DNN; in Figure 2.1.1.2 (c), the network is pruned from the top down, starting from the top layer; and in Figure 2.1.1.2 (d), unimportant filters within a single layer are removed to optimize the network further.

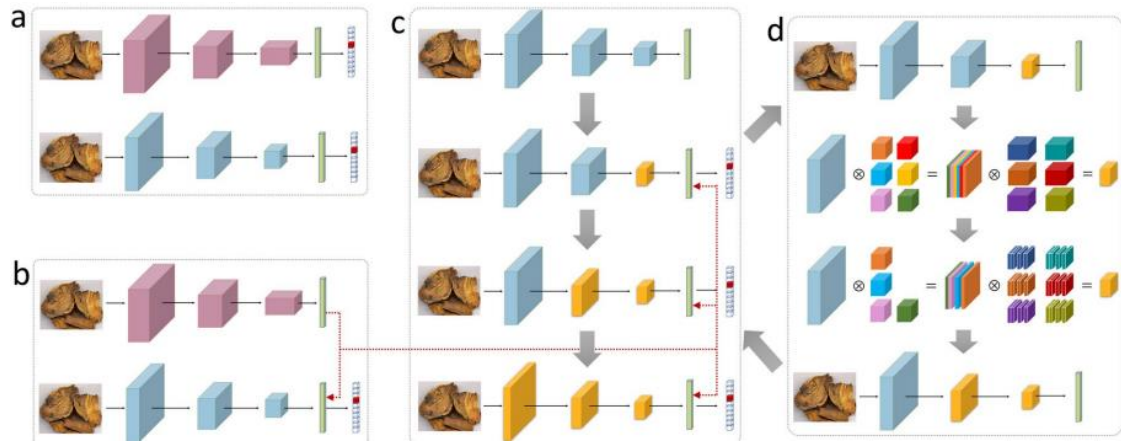


Figure 2.1.1. 2: Overview of the network compression approach [3].

The selection of methods in this paper is driven by the need to balance accuracy with efficiency. Traditional large deep learning models are not feasible for mobile devices due to their computational demands. The compression techniques used, including network transfer and network cutting, ensure that the final model remains accurate while being small enough to run efficiently on smartphones. This approach allows the application to be used in resource-limited settings without relying on expensive hardware.

The compressed deep learning model achieved a top 1 accuracy of 69.98% and a top-5 accuracy of 87.71% in Table 2.1.1.1, which is competitive compared to larger models but with significantly reduced computational requirements. The application demonstrated near-real-time performance on common smartphones, achieving up to 21 frames per second (fps) on higher-end models, making it practical for real-world use.

Table 2.1.1. 1: Performance evaluation of three smartphones [3].

Smartphone		HUAWEI Mate9 Pro	Xiaomi MI 9	HUAWEI P30 Pro
CPU type		Kirin 960	Qualcomm 855	Kirin 980 + NPU
Basic frequency		2.36 GHz	2.83 GHz	2.60 GHZ
RAM size		4 GB	8 GB	8 GB
Android version		9	9	10
Screen resolution		1080 x 1920	1080 x 2340	1080 x 2340
Large DNN	Average running time	342 ms	255 ms	187 ms
	Frames per second	2.9 fps	3.9 fps	5.3 fps
	Top1 accuracy	70.61 \pm 1.00		
	Top5 accuracy	88.97 \pm 0.41		
Small DNN + transfer	Average running time	130 ms	73 ms	59 ms
	Frames per second	7.7 fps	13.6 fps	16.7 fps
	Top1 accuracy	70.97 \pm 0.50		
	Top5 accuracy	88.72 \pm 0.79		
Small DNN + transfer + cut	Average running time	75 ms	51 ms	48 ms
	Frames per second	13.4 fps	19.6 fps	21.0 fps
	Top1 accuracy	69.98 \pm 1.09		
	Top5 accuracy	87.71 \pm 0.73		

In conclusion, this paper provides an effective solution for mobile herb image recognition using deep learning. By employing network compression techniques, the researchers managed to create a model that balances accuracy and efficiency, making it suitable for deployment on mobile devices.

2.1.2 CCNNet: a novel lightweight convolutional neural network and its application in traditional Chinese medicine recognition

Gang Hu et.al. [4] introduces a novel lightweight convolutional neural network (CNN) called CCNNet, specifically designed for traditional Chinese medicine (TCM) recognition. The motivation behind this research is the need for efficient and accurate models that can be deployed on edge devices with limited computational resources, such as mobile phones, to perform TCM recognition tasks. With the increasing role of TCM in healthcare, particularly highlighted during the neo-coronavirus epidemic, the demand for automated and intelligent TCM recognition tools has grown.

The development of lightweight CNNs is driven by the need to deploy deep learning models on mobile and edge devices that have limited processing power and memory. Traditional deep learning models, while powerful, are often too resource-intensive for such platforms. The proposed CCNNet aims to balance the trade-off between model complexity, accuracy, and computational efficiency, making it suitable for real-world applications on mobile devices.

The study introduces a new dataset called TCM-100 [4], created to address the lack of publicly available datasets for TCM image recognition. This dataset comprises 100 categories of TCM images, totalling over 60,000 images. These images include both self-collected photos and publicly available images from the internet. The dataset features fine-grained classification challenges, making it suitable for evaluating the performance of deep learning models like CCNNet.

The core of the paper's contribution is the design of CCNNet in Figure 2.1.2.1, which compresses modern CNN architectures while maintaining high accuracy. The model utilizes a bottleneck architecture combined with a Group Convolution Inverted Residual (GCIR) module and a Multi-Dimension Channel Attention (MDCA) mechanism to enhance performance. CCNNet compresses the network's architecture into three stages, reducing the number of parameters and computational load while preserving the ability to capture multi-scale features.

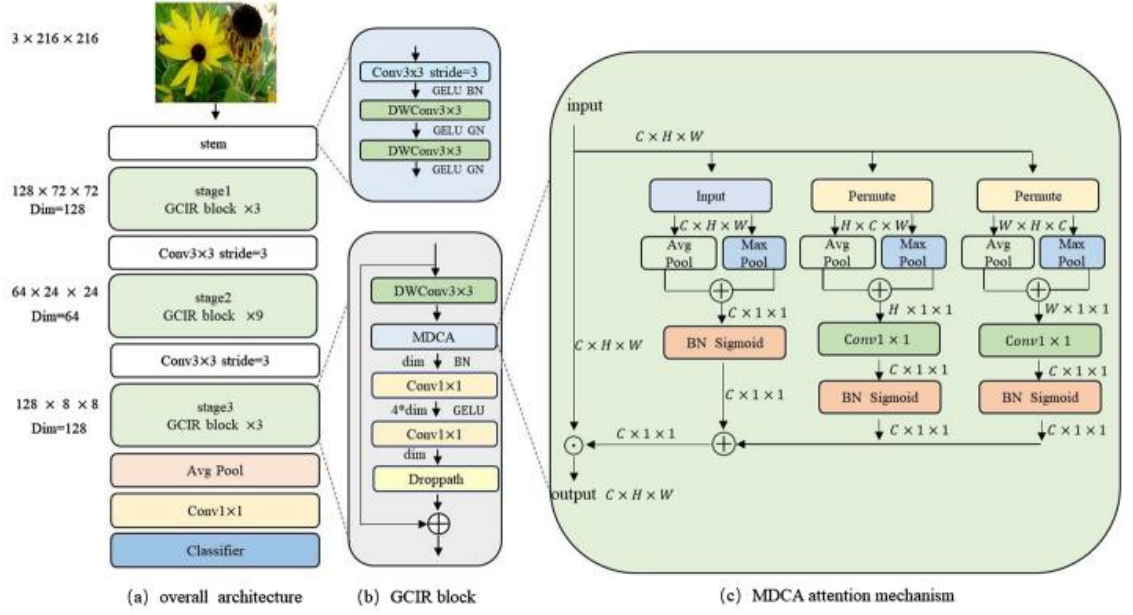


Figure 2.1.2. 1: Architecture of CCNNet [4].

The authors selected the lightweight design of CCNNet to address the need for high-performance models that can operate efficiently on mobile devices [4]. The GCIR module and MDCA attention mechanism were integrated to ensure that the model could capture both local and global features effectively, which is crucial for handling the variability in TCM images. The combination of these techniques results in a model that is both compact and powerful, making it suitable for edge device deployment.

CCNNet was tested on several datasets, including CIFAR-10, CIFAR-100, ImageNet-1K, and the newly created TCM-100. The model achieved a top-1 accuracy of 70.1% on ImageNet with only 1.4 million parameters in Table 2.1.2.1, outperforming several baseline models like MobileNetV3 and ShuffleNetV2. On the TCM-100 dataset, the expanded version of CCNNet (CCNNet3.0X) achieved a classification accuracy of 92.5% in Table 2.1.2.2, demonstrating its effectiveness in fine-grained TCM image recognition tasks.

Table 2.1.2. 1: Performance comparison of baseline models [4].

Model	Parameters/M	Flops/B	Top-1 acc		
			ImageNet (%)	cifar-10 (%)	cifar-100 (%)
MobileNetV3small	2.5	0.3	67.4	89.6	74.2
Shuffle NetV2 0.5X	1.4	0.15	61.1	88.9	71.4
GhostNet 0.5X	2.6	0.14	66.2	89.8	74.1
MobileNeXt 0.5X	1.8	0.3	67.7	89.9	75.2
MobileVit-XXS	1.3	0.7	69.0	90.2	76.3
CCNNet(Ours)	1.4	0.36	70.1	90.8	77.6

Table 2.1.2. 2: Evaluation of CCNNet expanded models across four datasets [4].

Model	Parameters/M	Flops/B	Top-1 accuracy			
			ImageNet (%)	cifar-10 (%)	cifar-100 (%)	TCM-100 (%)
CCNNet(1.0X)	1.4	0.36	70.1	90.8	77.6	86.8
CCNNet(1.5X)	2.1	0.46	73.2	91.9	79.4	87.3
CCNNet(2.0X)	3.1	0.55	75.1	93.8	80.1	89.2
CCNNet(3.0X)	5.4	0.74	76.3	94.7	81.6	92.5

This study provides critical insights into designing lightweight CNN models optimized for mobile deployment. CCNNet's ability to deliver high accuracy with minimal computational resources aligns well with the goals of the project. The methods and techniques discussed in the paper, such as model compression and the use of attention mechanisms, can be directly applied or adapted to improve the performance and efficiency of your mobile herb recognition application.

In summary, this paper presents an innovative approach to designing lightweight CNNs for TCM recognition, offering a model that balances high accuracy with computational efficiency. The proposed CCNNet model is well-suited for deployment on mobile devices, making it a relevant and valuable reference.

2.2 Limitations of Previous Studies

Previous studies on mobile herb and traditional Chinese medicine (TCM) recognition have faced significant limitations. One of the primary challenges is the inherent difficulty of balancing model efficiency with accuracy. In the first study, while network

compression effectively reduces the model size, it potentially compromises accuracy. Similarly, the second study's lightweight CCNNet, designed for efficiency, may underfit complex, fine-grained classification tasks due to its reduced model complexity.

2.3 Existing Mobile Applications

2.3.1 Herbapp

Herbapp [6] is an application designed for searching and identifying traditional Chinese herbs. It was released on June 7, 2023, and is offered by a developer named Watashi.

After logging in, users are directed to a page displaying all available herbs and formulas. At the top, there is a search function and a filter option to refine the selection of specific herbs or formulas. Users can also scroll down to explore and select the herbs or formulas they wish to learn more about. Once a herb is selected, the app provides detailed general information, including cautions, contraindications, flavors, dosage, channels, exams, actions, and symptoms. The 'Cautions' section warns of potential issues, such as problems that may arise from overconsumption. The 'Contraindications' section lists situations where the herb should not be taken, such as during pregnancy. This page also provides guidance on approximate dosage, benefits, related symptoms, and even includes exams for users to study the herb in depth. In addition, on the second page, information is provided about whether the selected herb interacts with other herbs or foods. If users are interested in learning about herb combinations, they can navigate to the third page and select a formula.

The formula page is quite like the herb page, with the main differences being the inclusion of tongue and pulse diagnostics, as well as examples of biomedical indications. The second page lists the herbs contained in the formula, categorized into chief herbs, deputy herbs, assistant herbs, and envoy herbs. This app also includes various flashcard decks with different types of practice questions designed for learning about herbs. Figure 2.3.1.1 to Figure 2.3.1.7 is the interface in the application.

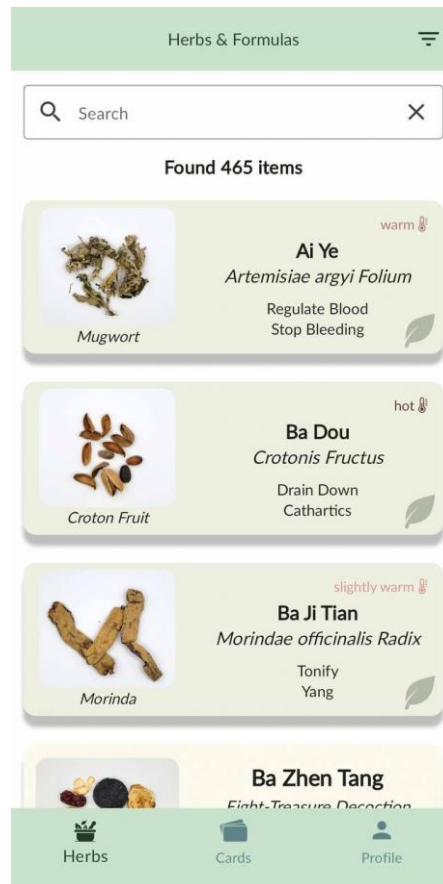


Figure 2.3.1. 1: Main Page of Hearbapp.

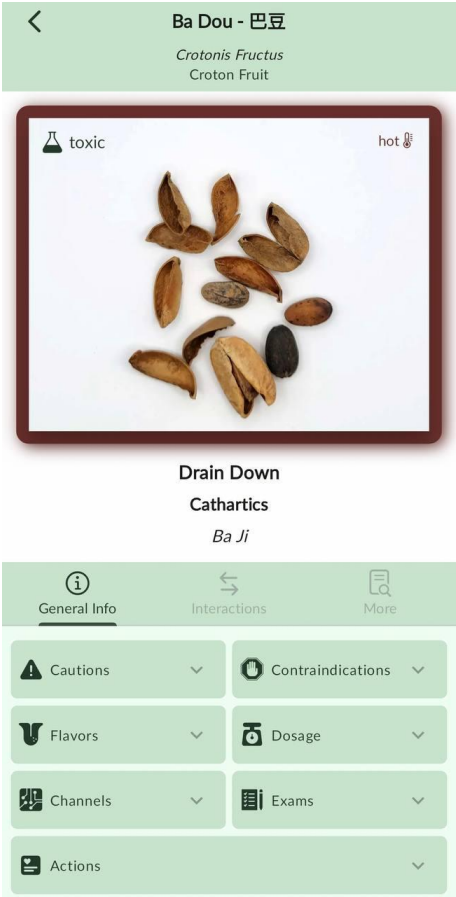


Figure 2.3.1. 2: Page of Herb.

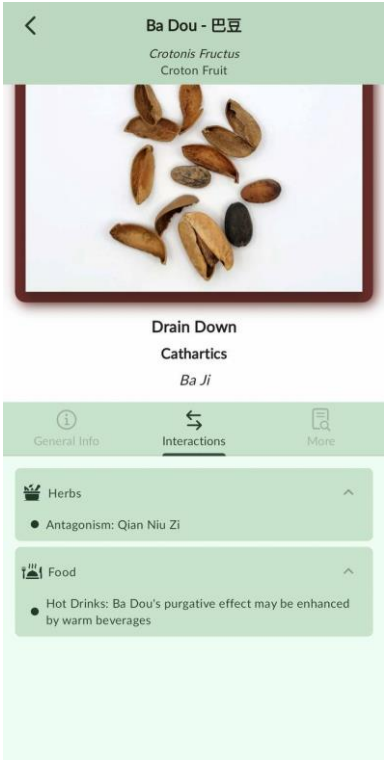


Figure 2.3.1. 3: Page of interactions in herb.

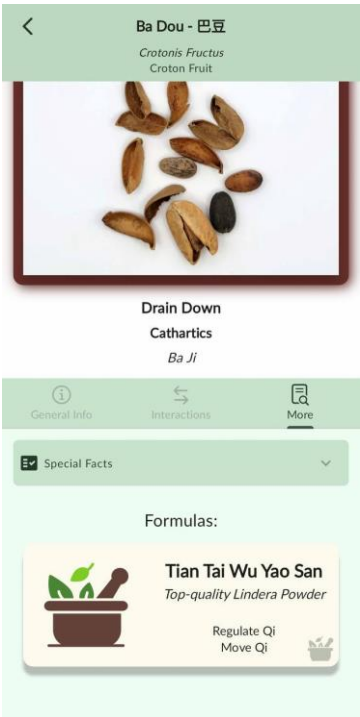


Figure 2.3.1. 4: Page of related formulas of the herb.

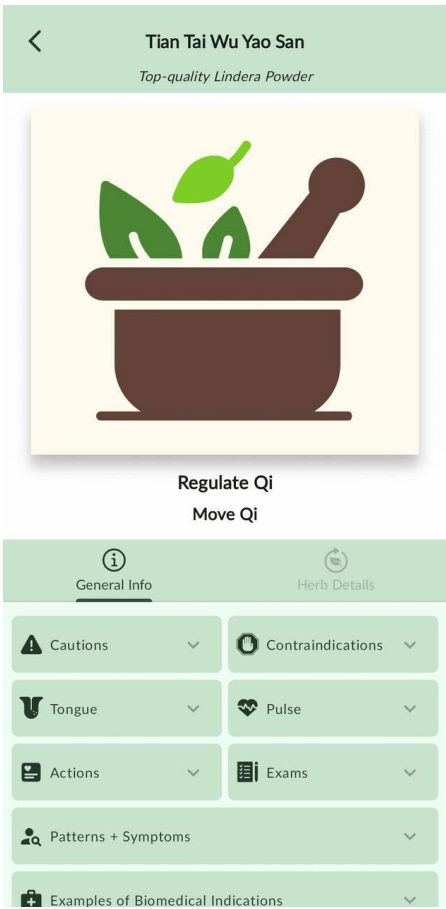


Figure 2.3.1. 5: Page of formula.



Figure 2.3.1. 6: Page of herbs in formula.



Figure 2.3.1. 7: Page of flash card decks.

Strength

- The app offers detailed information on a wide range of herbs
- The app has search and filter functions, along with the organized layout of herbs and formulas, allow users to quickly find and access the information they need
- The app has flashcard decks with practice questions enhances the learning experience, making it easier to memorize the properties of various herbs.
- Information on interactions between herbs and foods adds an extra layer of safety and utility, helping users avoid potential complications.

Weakness

- Lack of practical guidance on how to apply this knowledge in real-life situations, such as self-preparation of formulas or dosage adjustments.
- After a certain period or number of uses, the app may require a subscription or charge fees to continue using its full range of features, which could be a drawback for users looking for a free resource.

2.3.2 Traditional Chinese Medicine

Traditional Chinese Medicine [7] is a simplified Traditional Chinese Medicine (TCM) app. Upon entering, it displays all categories of medicinal drugs, such as hemostatic drugs, expectorant drugs, and cough suppressants. Users can select a category based on their symptoms. After selecting a category, information about the types of drugs included is displayed, along with the herbs that make up each drug. By clicking on these herbs, detailed information is provided, including the herb's origin, efficacy, characteristics, processing methods, and examples of formulas containing the herb. The app has limited functionality, and unlocking more features requires an upgrade through a paid subscription. Figure 2.3.2.1 to Figure 2.3.2.3 is the interface in the application.



Figure 2.3.2. 1: Main page of Traditional Chinese Medicine.



Figure 2.3.2. 2: Information about the types of drugs.

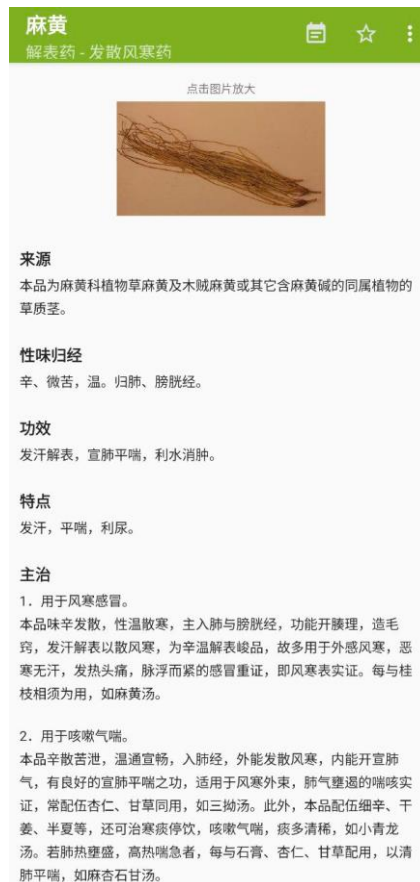


Figure 2.3.2. 3: Information of the herb.

Strength

- Users can quickly find relevant drug categories based on their symptoms, making it convenient to identify potential remedies.
- Each herb includes detailed information such as origin, efficacy, characteristics, processing methods, cooking instructions, and usage, as well as examples of formulas. This provides users with a thorough understanding of how to prepare and use the herbs effectively.

Weakness

- While the app is simplified, unlocking more advanced features requires a paid subscription, which might deter users looking for a free resource.
- The user interface design is not well-optimized, leading to a less-than-ideal user experience, which may affect overall usability and satisfaction.

2.3.3 TCM Clinic Aid Trial

TCM Clinic Aid Trial [8] is a TCM (Traditional Chinese Medicine) application with several key features. Users can find herbs based on symptoms or directly search for specific herbs they wish to learn about. The homepage displays all available functions, allowing users to choose between whole word or partial word searches. Searches can also be refined by specific fields or points. Additionally, users can search by symptoms, diagnosis, materia medica, or acupuncture points, and create a favorites list to bookmark items.

The app also offers quizzes to test knowledge about herbs. Within the symptoms section, users can view related formulas, herbs, and points based on the selected symptom. The app provides comprehensive details for each herb, including their functions, combinations, and other relevant information. However, a significant drawback is that many of the app's functions are unusable, which greatly impacts the overall user experience. Figure 2.3.3.1 to Figure 2.3.3.5 is the interface in the application.

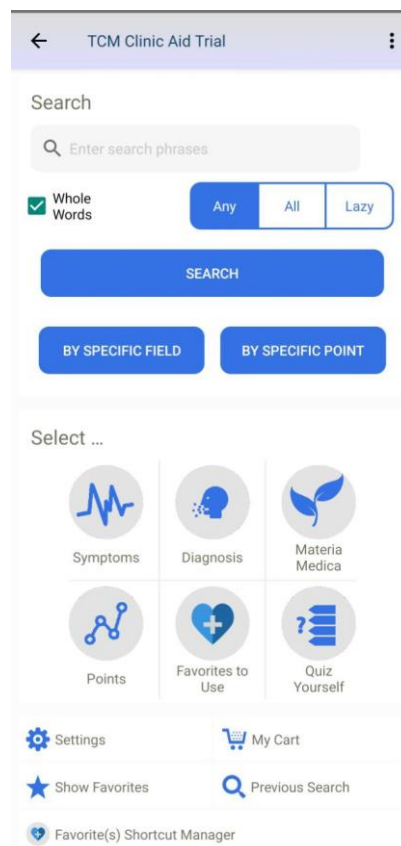


Figure 2.3.3. 1: Main page of TCM Clinic Aid Trial.

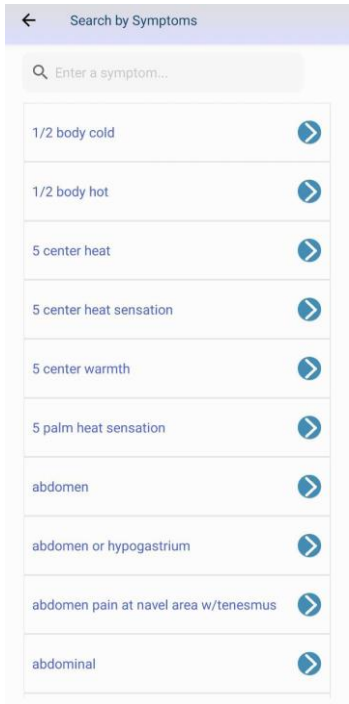


Figure 2.3.3. 2: Search by Symptoms.

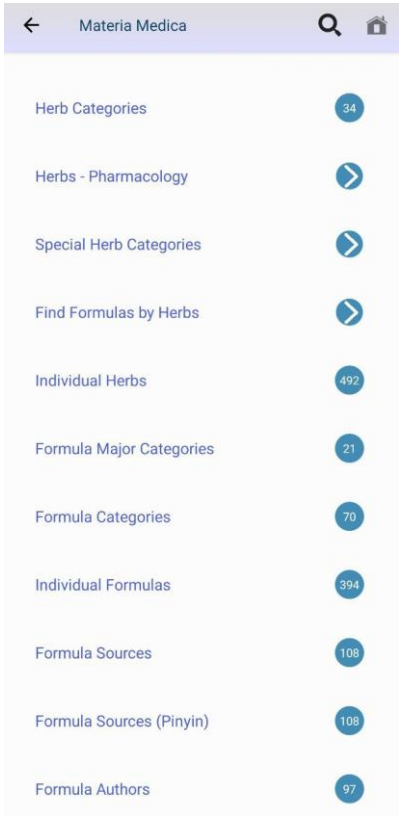


Figure 2.3.3. 3: Search by materia medica.

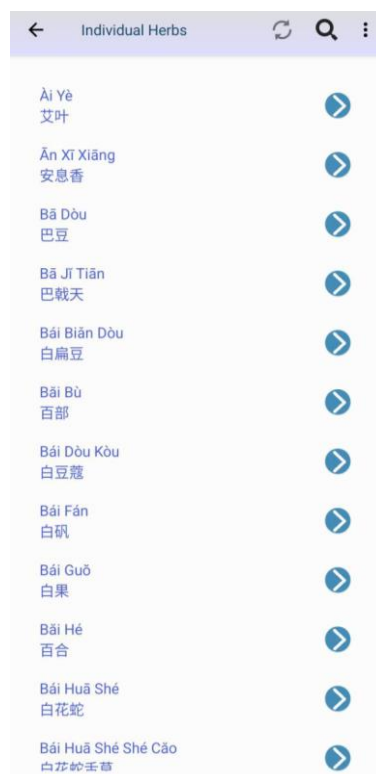


Figure 2.3.3. 4: Page of Individual Herbs.

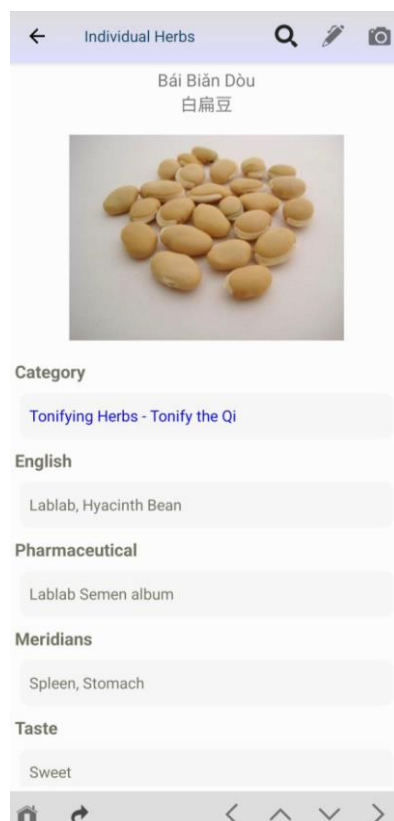


Figure 2.3.3. 5: Information of herbs.

Strength

- The ability to create a favourites list enables users to bookmark and quickly access frequently used herbs or formulas.
- The app allows for detailed searches by whole word, partial word, specific fields, or points, making it versatile for different user needs.
- The app includes quizzes that allow users to test and reinforce their knowledge of herbs, making it a useful educational tool.

Weakness

- The user interface design is not well-optimized, leading to a less-than-ideal user experience, which may affect overall usability and satisfaction.
- A significant number of the app's functions are unusable, severely impacting its effectiveness and user experience.
- The app does not provide a smooth user experience, with navigation and interactions feeling clunky and unintuitive.
- The app's functions are not clearly explained, making it difficult for users to understand how to use certain features effectively.

2.4 Summary

The literature review explores advancements in the recognition of TCM through deep learning techniques, focusing on two significant studies. The first study by Xin Sun et al. [3] presents a mobile application utilizing deep learning for herb image recognition, addressing the inefficiencies of traditional methods reliant on expert knowledge and chemical processes. The researchers developed a lightweight deep learning model that can efficiently run on smartphones, using techniques like network compression to balance accuracy and efficiency. The second study by Gang Hu et al. [4] introduces CCNNet, a novel lightweight convolutional neural network designed specifically for TCM recognition on mobile and edge devices with limited computational resources. The study addresses the growing demand for automated TCM recognition tools, especially in the context of healthcare during the neo-coronavirus epidemic. CCNNet, tested on several datasets including a newly created TCM-100 dataset, achieved high

accuracy with minimal computational requirements. The model's lightweight design, incorporating advanced techniques like the Group Convolution Inverted Residual module and Multi-Dimension Channel Attention mechanism, makes it highly effective for mobile deployment, offering a balance between model complexity, accuracy, and computational efficiency.

Three existing applications are listed out strengths and weaknesses and summarised and concluded in Table 2.4.1.

Table 2.4. 1: Summary of strengths and weaknesses.

	Hearbapp	Traditional Chinese Medicine	TCM Clinic Aid Trial
Detailed Information	Offers detailed information on a wide range of herbs but no cooking instructions	Offers detailed information on a wide range of herbs and have cooking instructions	Offers detailed information on a wide range of herbs but no cooking instructions
Educational Tools	Flashcard decks with practice questions	-	Quizzes to test
Favourites List	-	Need Subscription	Have Favourites List and free to use
User Interface (UI) Design	well	Not well	Not well
Subscription/Fees	May require a subscription or charge fees after a certain period or number of uses	Unlocking more advanced features requires a paid subscription	-

CHAPTER 3 – System Design

3.1 Use-Cases Diagram

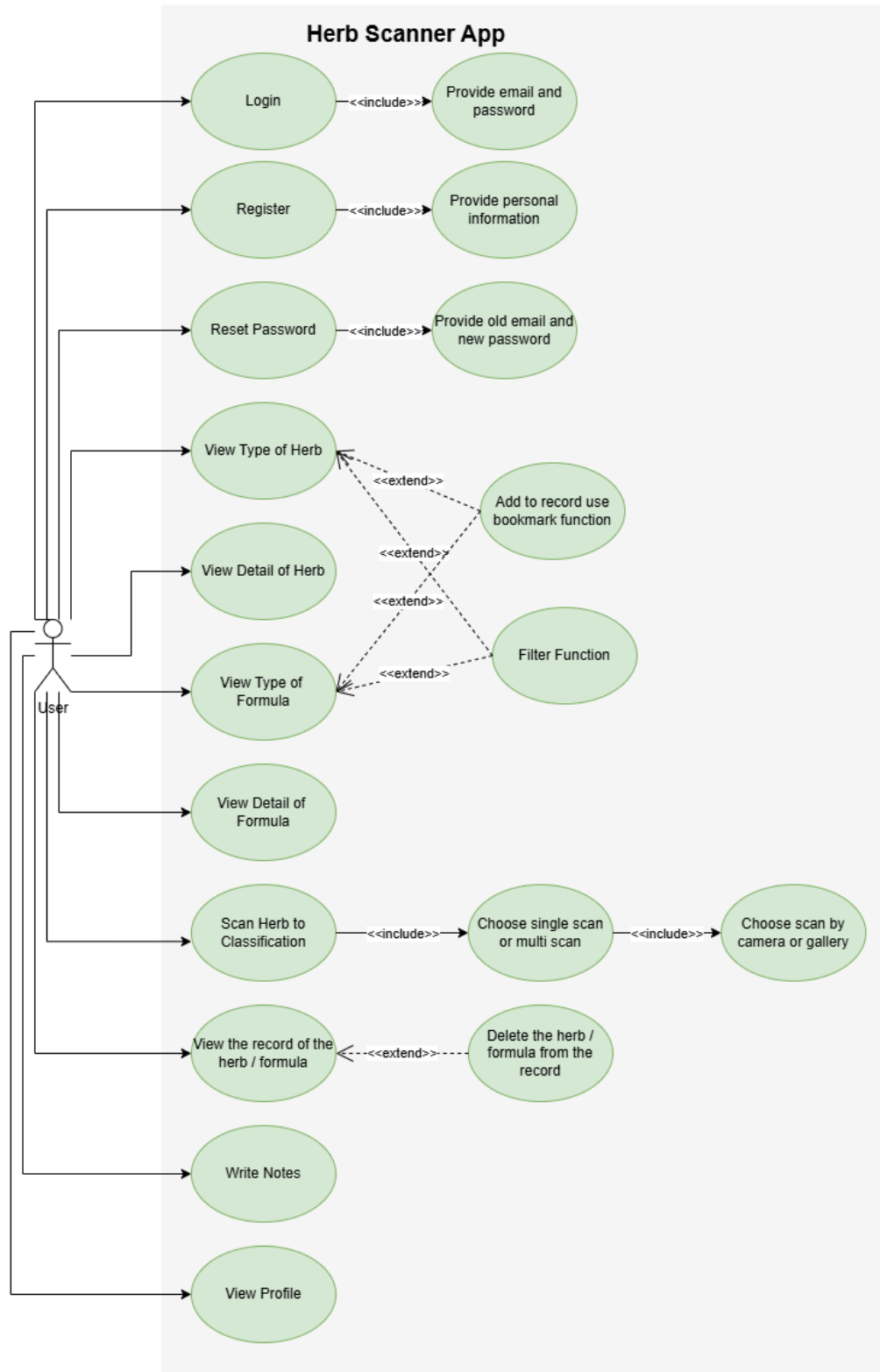


Figure 3.1. 1: Use-cases diagram.

3.2 Use-case Description

Table 3.2. 1: Login use case description.

Use case ID	001
Use case Name	Login
Brief Description	To allow user access the app
Actor	User
Trigger	First page of the app
Precondition	Open the app
Normal flow of event	<ol style="list-style-type: none"> 1. User key in their email and password or use their own google account to log in 2. System with check the email and password exists and correct or not 3. Direct to the herb type page
Sub flows	<ol style="list-style-type: none"> 1. Users need login every time when open the app
Alternate flows	<ol style="list-style-type: none"> 1. If password is wrong, will alert an error message 2. If google account is no exists will automatically create a new account

Table 3.2. 2: Register use case description.

Use case ID	002
Use case Name	Register
Brief Description	To allow new user register account
Actor	User
Trigger	Click the register link in login page
Precondition	The email has not been registered yet
Normal flow of event	<ol style="list-style-type: none"> 1. Click the “register here” below the login page 2. Fill in the information

	<ol style="list-style-type: none"> 3. Click the “register” button 4. Direct back to the login page automatically
Sub flows	-
Alternate flows	<ol style="list-style-type: none"> 1. If password is less than 6 characters, system will alert register failed 2. If email is already registered, system will alert register failed

Table 3.2. 3: Reset passwords use case description.

Use case ID	003
Use case Name	Reset password
Brief Description	To allow user reset password if they forget their password
Actor	User
Trigger	Click the “Forgot Password” link in login page
Precondition	The email must already register before
Normal flow of event	<ol style="list-style-type: none"> 1. User key in their email 2. Click the “Forgot Password” link 3. System will send the reset password link to their email 4. User key in their new password and confirm their new password to reset
Sub flows	-
Alternate flows	<ol style="list-style-type: none"> 1. If user no key in their email and direct click the “Forgot Password” link, system will alert error message

Table 3.2. 4: View type of herb use case description.

Use case ID	004
Use case Name	View type of herb
Brief Description	To allow user to view the type of herb
Actor	User
Trigger	Click the login button in login page
Precondition	Successfully login
Normal flow of event	1. Scroll down the page to view the herb
Sub flows	1. User can search the herb's name in the search bar 2. User can filter by herb's name or properties 3. User can add bookmarks to record
Alternate flows	-

Table 3.2. 5: View detail of herb use case description

Use case ID	005
Use case Name	View detail of herb
Brief Description	To allow user to view the information of herb
Actor	User
Trigger	Click the herb card to view type of herb page
Precondition	-
Normal flow of event	1. Scroll down the page to view the information of herb
Sub flows	-
Alternate flows	-

Table 3.2. 6: View type of formula use case description.

Use case ID	006
Use case Name	View type of formula
Brief Description	To allow user to view the type of formula
Actor	User
Trigger	Click the login button in login page
Precondition	Successfully login
Normal flow of event	1. Scroll down the page to view the formula
Sub flows	1. User can search the formula's name or symptoms in the search bar 2. User can filter by formula's name or properties 3. User can add bookmarks to record
Alternate flows	-

Table 3.2. 7: View detail of formula use case description.

Use case ID	007
Use case Name	View detail of formula
Brief Description	To allow user to view the information of formula
Actor	User
Trigger	Click the formula card in view type of herb page
Precondition	-
Normal flow of event	1. Scroll down the page to view the information of formula

Sub flows	1. User can click the herb in the combination to view the herb's information
Alternate flows	1. If an herb in the formula combination is not found in the herb list, the system shows: "Herb not found."

Table 3.2. 8: Scan herb use case description.

Use case ID	008
Use case Name	Scan herb to classification
Brief Description	To allow user to classification the herb using camera or gallery
Actor	User
Trigger	Click "Camera" button at the bottom of the page
Precondition	User need give the permission of the camera
Normal flow of event	<ol style="list-style-type: none"> 1. User navigates to the Herb Scanner page. 2. User selects Scan Mode: either Scan Single or Scan Multi. 3. User chooses input method: <ul style="list-style-type: none"> • Tap Scan to open the camera • Or tap Gallery to choose an image 4. System sends the image to corresponding model (EfficientNet or YOLO) 5. The classification result pops up showing detected herb or formula

Sub flows	<ol style="list-style-type: none"> 1. User taps on detected herb to open the herb detail page in Single Scan. 2. In Multi Scan, multiple herbs and formula are displayed in result cards. 3. After viewing, user may return to scanner to perform another scan. 4. User can switch between Scan Single and Scan Multi modes.
Alternate flows	<ol style="list-style-type: none"> 1. User selects “Gallery” instead of using the camera to upload an image. 2. System fails to detect any herb and shows a message: “No detection. Try another image.” 3. Network or API error occurs and system shows: “Connection failed. Please check your internet.” 4. User has not granted camera permission, and system shows: “Camera access is required.” 5. In Multi Scan mode, if the detected herbs do not match any formula, the system only displays the list of detected herbs without formula suggestion.

Table 3.2. 9: View the record use case descriptions.

Use case ID	009
Use case Name	View the record of the herb and formula
Brief Description	To allow user to view the bookmark of the herb and formula
Actor	User
Trigger	Click “Record” button at the bottom of the page
Precondition	-
Normal flow of event	<ol style="list-style-type: none"> 1. User clicks the “Record” button on the bottom navigation bar. 2. The “Saved Items” page opens with two tabs: Herbs and Formulas. 3. User can scroll through the list to view saved herbs and formulas.
Sub flows	<ol style="list-style-type: none"> 1. User can switch between the “Herbs” and “Formulas” tabs. 2. User clicks on a saved herb or formula to view detailed information. 3. User can remove a herb or formula from the saved list.
Alternate flows	-

Table 3.2. 10: Write notes use case descriptions.

Use case ID	0010
Use case Name	Write notes
Brief Description	To allow user to write the notes
Actor	User

Trigger	Click “Notes” button at the bottom of the page
Precondition	-
Normal flow of event	<ol style="list-style-type: none"> 1. User clicks the “Notes” button in the bottom navigation bar. 2. The “My Notes” page is displayed, listing all saved notes. 3. User clicks the “+” button to create a new note. 4. User types the content and saves the note. 5. The new note appears in the list.
Sub flows	<ol style="list-style-type: none"> 1. User can click an existing note to view and edit its content. 2. User can delete a note by clicking the trash icon.
Alternate flows	<ol style="list-style-type: none"> 1. If no notes are saved, the system displays: “No notes available.”

Table 3.2. 11: View profile use case description

Use case ID	0011
Use case Name	View Profile
Brief Description	To allow user to review their name and app version on the profile page. The profile image depends on the login method.
Actor	User
Trigger	Click “Profile” button at the bottom of the page
Precondition	-

Normal flow of event	<ol style="list-style-type: none"> 1. Click “Profile” button at the bottom of the page 2. User can review their name
Sub flows	<ol style="list-style-type: none"> 1. User can log out the account in profile page
Alternate flows	<ol style="list-style-type: none"> 1. If the user logs in using Google, their Google profile picture will be displayed in the profile page. 2. If the user logs in using email and password, no profile picture will be shown (default avatar is used).

3.3 Program Development

3.3.1 Login Page Development

The login page provides three key authentication functionalities: email-password login, Google Sign-In, and password reset. The `_login` function in Figure 3.3.1.1 uses Firebase's `signInWithEmailAndPassword` to authenticate users with their email and password, displaying success or failure messages accordingly. The `_signInWithGoogle` function in Figure 3.3.1.2 integrates Google Sign-In, allowing users to log in via their Google account credentials, which are authenticated with Firebase. Lastly, the `_resetPassword` function in Figure 3.3.1.3 enables users to reset forgotten passwords by sending a password reset email through Firebase.

To enhance user experience, the app uses a `saveLoginStatus()` function in Figure 3.3.1.4 to store the user's login state using `SharedPreferences`. Once the user has logged in successfully, their login information is saved locally. This allows the app to automatically keep the user signed in on subsequent launches, unless the user explicitly logs out. These features together ensure a user-friendly, persistent, and secure authentication system for the app.

```

Future<void> _login() async {
  try {
    await _auth.signInWithEmailAndPassword(
      email: emailController.text,
      password: passwordController.text,
    );
    await saveLoginStatus(); //record login status
    Navigator.pushReplacement(context, MaterialPageRoute(builder: (_) => MainPage()));
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text("Login failed: $e")));
  }
}

```

Figure 3.3.1. 1: Code of email sign in function.

```

Future<void> _signInWithGoogle() async {
  try {
    final googleUser = await _googleSignIn.signIn();

    if (googleUser == null) return;

    final googleAuth = await googleUser.authentication;

    final credential = GoogleAuthProvider.credential(
      accessToken: googleAuth.accessToken,
      idToken: googleAuth.idToken,
    );

    await _auth.signInWithCredential(credential);

    await saveLoginStatus(); //record login status

    Navigator.pushReplacement(context, MaterialPageRoute(builder: (_) => MainPage()));
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text("Google Sign-In failed: $e")));
  }
}

```

Figure 3.3.1. 2: Code of google sign in function.

```

Future<void> _resetPassword() async {
  final email = emailController.text.trim();

  if (email.isEmpty) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text("Please enter your email address.")),
    );
    return;
  }

  try {
    await _auth.sendPasswordResetEmail(email: email);
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text("Password reset email sent!")),
    );
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text("Failed to send reset email: $e")),
    );
  }
}

```

Figure 3.3.1. 3: Code of reset password function.

```

Future<void> saveLoginStatus() async {
  final prefs = await SharedPreferences.getInstance();
  await prefs.setBool('isLoggedIn', true);
}

```

Figure 3.3.1. 4: Code of save login status.

3.3.2 Register Page Development

This Register Page provides a user registration functionality for the app, as shown in Figure 3.3.2.1 and Figure 3.3.2.2. It allows users to create an account by entering their name, email, and password. The registration process includes validation checks to ensure that all fields are filled and the email format is valid. If any input is invalid, appropriate error messages are shown using SnackBar.

Upon validation, the app uses Firebase Authentication to create a new user account with the provided email and password. At the same time, the user's name and email are stored in Firestore under the users collection, allowing for future reference and personalization.

A loading state is triggered using the `_isLoading` flag to indicate the process is ongoing, providing visual feedback. Once registration is successful, the app displays a confirmation message using `SnackBar` and redirects the user to the login page via `Navigator.pushReplacement`.

```
Future<void> _register() async {
  if (nameController.text.trim().isEmpty || emailController.text.trim().isEmpty || passwordController.text.isEmpty) {
    ScaffoldMessenger.of(context).showSnackBar(const SnackBar(content: Text('Please fill in all fields')));
    return;
  }

  if (!IsValidEmail(emailController.text.trim())) {
    ScaffoldMessenger.of(context).showSnackBar(const SnackBar(content: Text('Please enter a valid email')));
    return;
  }

  setState(() {
    _isLoading = true;
  });
  try {
    UserCredential userCredential = await _auth.createUserWithEmailAndPassword(
      email: emailController.text,
      password: passwordController.text,
    );

    await _firestore.collection('users').doc(userCredential.user?.uid).set({
      'name': nameController.text,
      'email': emailController.text,
    });
  }
}
```

Figure 3.3.2. 1: Code of register function.

```
    ScaffoldMessenger.of(context).showSnackBar(const SnackBar(content: Text("Registration successful!")));
    Navigator.pushReplacement(context, MaterialPageRoute(builder: (_) => LoginScreen()));
  } catch (e) {
    print("Registration Error: $e");
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(content: Text("Registration failed: $e")));
  } finally {
    setState(() {
      _isLoading = false;
    });
  }
}
```

Figure 3.3.2. 2: Code of register function 2.

3.3.3 Herb type Page Development

The herb section is implemented using the HerbsPage and HerbCard components. As shown in Figure 3.3.3.1, HerbsPage dynamically loads herb data from a local JSON file using the rootBundle method. The dataset, illustrated in Figure 3.3.3.2, was manually compiled from reliable online resources such as Baidu and organized into a standardized format for efficient parsing.

The page features a responsive search bar and quick filter options. Users can search herbs by Chinese name, English name, Latin name, or specific indications. The keyword-based search logic is implemented in the _visibleItems getter, as shown in Figure 3.3.3.3, where matching is applied across herb and formula fields.

In addition, a quick filter interface is provided to refine herb results based on properties such as warm, cold, hot, using FilterChip widgets. This user-friendly UI is demonstrated in Figure 3.3.3.4, improving accessibility and allowing users to narrow down herbs by nature.

```
Future<void> _loadData() async {
  final herbData = await rootBundle.loadString('assets/all_herbs.json');
  final formulaData = await rootBundle.loadString('assets/herb_formulas.json');

  final List<dynamic> herbJson = json.decode(herbData);
  final Map<String, dynamic> formulaJson = json.decode(formulaData);

  setState(() {
    _herbs = herbJson.map((h) => Herb.fromJson(h)).toList();
    _formulas = (formulaJson['formulas'] as List).map((f) => Formula.fromJson(f)).toList();
  });
}
```

Figure 3.3.3. 1: Load herb data from Herb JSON file function.

```

{
  "herb_ID": "herbID_00034",
  "cn_name": "川芎",
  "eng_name": "Chuanxiong Rhizome",
  "latin_name": "Rhizoma Chuanxiong",
  "description": [
    "川芎为伞形科植物川芎Ligusticum chuanxiong Hort.的干燥根茎。",
    "具有活血行气、祛风止痛的作用，是治疗血瘀气滞及头痛的重要药材。"
  ],
  "properties": "辛，温",
  "usePart": "干燥根茎",
  "functions": [
    {"content": "活血行气"},
    {"content": "祛风止痛"}
  ],
  "indication": [
    "血瘀经闭",
    "胸胁胀痛",
    "风湿痹痛",
    "头痛眩晕"
  ],
  "dosage": "3~9克",
  "Contraindication": "孕妇慎用，阴虚火旺者慎用",
  "Channel": [
    "肝经",
    "胆经",
    "心包经"
  ],
  "image_path": "assets/herb_images/chuanxiong.png"
},

```

Figure 3.3.3. 2: Herb JSON file.

```

if (_searchText.isNotEmpty) {
  items = items.where((item) {
    if (item is Herb) {
      return item.cnName.toLowerCase().contains(_searchText) ||
        item.engName.toLowerCase().contains(_searchText) ||
        item.latinName.toLowerCase().contains(_searchText);
    } else if (item is Formula) {
      return item.nameCn.toLowerCase().contains(_searchText) ||
        item.nameEn.toLowerCase().contains(_searchText) ||
        item.indications.any((i) => i.toLowerCase().contains(_searchText));
    }
    return false;
  }).toList();
}

```

Figure 3.3.3. 3: Code of herb search bar.


```

if (_selectedProperties.isNotEmpty) {
  items = items.where((item) {
    if (item is Herb) {
      return _selectedProperties.any((p) => item.properties.contains(p));
    }
    return false;
  }).toList();
}

```

Figure 3.3.3. 4: Code of herb filter by properties.

3.3.4 Formula type Page Development

The formula section is managed within the HerbsPage component and activated through category filtering, as described in Figure 3.3.4.1. When the "Formula" category is selected, the application presents formula data that has been preloaded from a structured local JSON file.

The JSON format, illustrated in Figure 3.3.4.2, contains key fields such as formula_id, name_cn, name_en, description, indications, and a list of herbs with their dosage and role. It also includes additional metadata such as preparation, contraindications, and note. The data was compiled manually from reliable TCM databases and follows a consistent schema.

Filtering is handled using the shared _visibleItems logic. In addition to text-based search by name or symptoms, a symptom-based filter enables targeted retrieval of formulas by checking whether selected symptoms appear in the formula's indications. This filtering logic is shown in Figure 3.3.4.3.

```

List<dynamic> items = [];
if (_selectedCategory == 'All') {
    items = [..._herbs, ..._formulas];
} else if (_selectedCategory == 'Herb') {
    items = [..._herbs];
} else if (_selectedCategory == 'Formula') {
    items = [..._formulas];
}

```

Figure 3.3.4. 1: Code of Category filter.

```

{
  "formula_id": "F001",
  "name_cn": "桂枝汤",
  "name_en": "Gui Zhi Decoction",
  "description": "用于治疗外感风寒、营卫不和引起的症状。",
  "indications": [
    "风寒感冒",
    "头痛",
    "发热",
    "鼻塞"
  ],
  "herbs": [
    {"name_cn": "桂枝", "role": "君药", "dosage": "9g"},
    {"name_cn": "芍药", "role": "臣药", "dosage": "9g"},
    {"name_cn": "炙甘草", "role": "佐药", "dosage": "6g"},
    {"name_cn": "生姜", "role": "佐药", "dosage": "9g"},
    {"name_cn": "大枣", "role": "使药", "dosage": "12枚"}
  ],
  "preparation": "1. 将所有药材（桂枝、芍药、炙甘草、生姜、大枣）洗净。",
  "contraindications": "表虚自汗者慎用。",
  "cautions": "孕妇慎用。",
  "pulse": "浮弱",
  "note": "调和营卫，解除风寒。",
  "image": "assets/formula_images/quizhitang.png"
},

```

Figure 3.3.4. 2: Formula JSON file.

```

if (_selectedSymptoms.isNotEmpty) {
  items = items.where((item) {
    if (item is Formula) {
      return _selectedSymptoms.any((symptom) =>
        item.indications.any((i) => i.contains(symptom)));
    }
    return false;
  }).toList();
}

```

Figure 3.3.4. 3: Code of formula filter by symptoms.

3.3.5 Herb Detail Page Development

The Herb Detail Page component is responsible for presenting detailed information about a selected herb. As shown in Figure 3.3.5.1, it includes sections such as Latin name, description, indications, properties, use part, dosage, contraindications, and associated meridians (channels). The page dynamically renders each section only if the corresponding data is valid and non-empty, skipping attributes marked as "NA" or blank.

Additionally, the page integrates a reusable helper method named `_buildFunctionCard`, which displays a list of functions associated with the herb. This is illustrated in Figure 3.3.5.2, where each function consists of a title and description, both extracted from a structured list. This approach ensures a clean and informative layout tailored to the available content of each herb.

All data shown is passed through a Herb object, enabling centralized formatting and logic handling. The implementation emphasizes flexibility and data-driven presentation while supporting bilingual display when applicable.

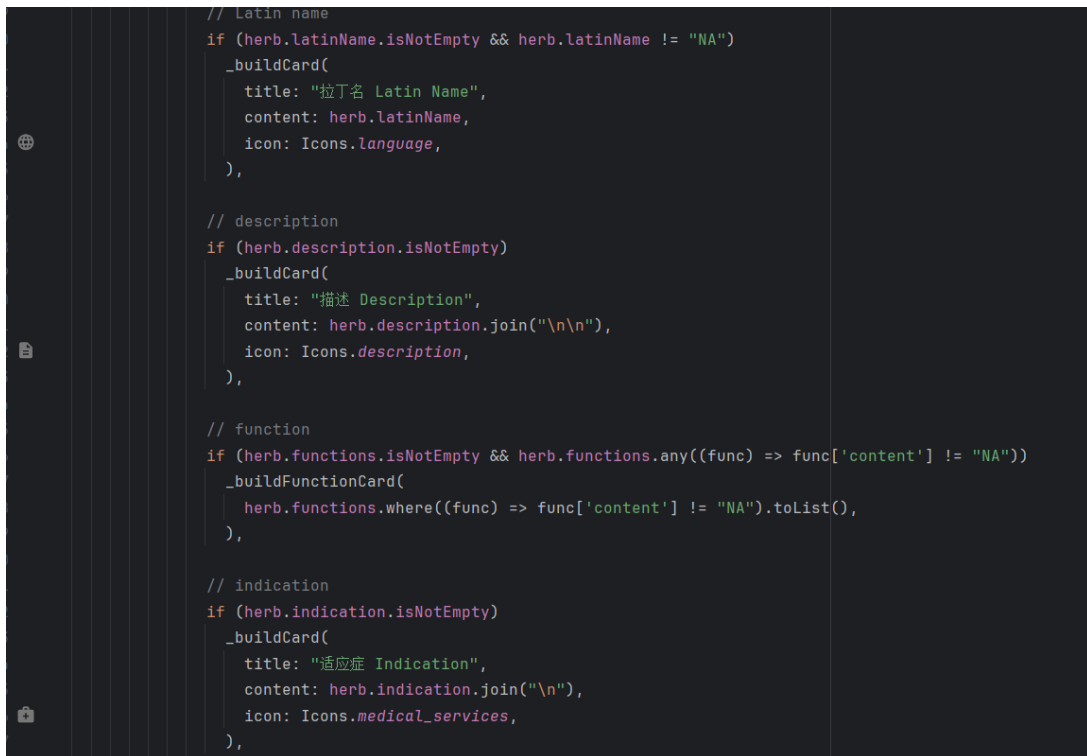


Figure 3.3.5. 1: Part of display herb detail section.

```

Widget _buildFunctionCard(List<Map<String, String>> functions) {
  return Card(
    color: Colors.green[50],
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.circular(12),
    ), // RoundedRectangleBorder
    elevation: 4,
    margin: const EdgeInsets.symmetric(vertical: 8),
    child: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          // title
          Row(
            children: [
              Icon(Icons.spa, color: Colors.green[800]),
              const SizedBox(width: 8),
              Container(
                decoration: BoxDecoration(
                  color: Colors.green[200],
                  borderRadius: BorderRadius.circular(8),
                ), // BoxDecoration
                padding:
                  const EdgeInsets.symmetric(horizontal: 8, vertical: 4),
                child: Text(
                  "功能 Functions",
                  style: TextStyle(
                    fontSize: 16,
                    fontWeight: FontWeight.bold,

```

Figure 3.3.5. 2: Function Card to display herb data.

3.3.6 Formula Detail Page Development

The Formula Detail Page component is used to display comprehensive information about a selected traditional Chinese medicine formula. As shown in Figure 3.3.6.1, the page presents structured fields such as the formula's Chinese and English names, description, indications, preparation method, contraindications, cautions, pulse characteristics, and notes. Each field is conditionally displayed based on its content to avoid showing empty or invalid data.

The ingredients section of the formula is organized using a role-based classification system (Chief, Deputy, Assistant, Envoy), extracted from the herbs field. These are grouped and presented through a helper function named `_buildHerbsCard`, as illustrated in Figure 3.3.6.2, allowing users to understand each herb's functional position within the formula.

A key interactive feature allows users to tap on any herb name to view its details. As shown in Figure 3.3.6.3, the system attempts to match the selected herb with entries in the main herb database and navigates to the corresponding `HerbDetailPage`. If no match is found, the system displays a dialog to inform the user. This ensures smooth navigation between formula and herb data, supporting effective information exploration.

```

if (formula.description.isNotEmpty)
  _buildCard(
    title: "描述 Description",
    content: formula.description,
    icon: Icons.description,
  ),
if (formula.indications.isNotEmpty)
  _buildCard(
    title: "适应症 Indication",
    content: formula.indications.join('\n'),
    icon: Icons.local_hospital,
  ),
if (formula.herbs.isNotEmpty)
  _buildHerbsCard(formula.herbs),
if (formula.preparation.isNotEmpty)
  _buildCard(
    title: "煮法 Preparation",
    content: formula.preparation,
    icon: Icons.kitchen,
  ),
if (formula.contraindications.isNotEmpty)
  _buildCard(
    title: "禁忌 Contraindications",
    content: formula.contraindications,
    icon: Icons.warning,
  ),

```

Figure 3.3.6. 1: Part of display formula detail.

```
Widget _buildHerbsCard(List<Map<String, String>> herbs) {
  Map<String, String> roleTranslations = {
    '君药': 'Chief',
    '臣药': 'Deputy',
    '佐药': 'Assistant',
    '使药': 'Envoy',
  };

  Map<String, List<Map<String, String>>> groupedHerbs = {
    '君药': [],
    '臣药': [],
    '佐药': [],
    '使药': [],
  };
};
```

Figure 3.3.6. 2: Code of herbs card.

```
return GestureDetector(
  onTap: () {
    final herbName = herb['name_cn']?.trim() ?? '';

    try {
      final matchedHerb = herbsList.firstWhere(
        (h) =>
          h.cnName == herbName ||
          h.engName.toLowerCase() == herbName.toLowerCase(),
      );

      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => HerbDetailPage(herb: matchedHerb),
        ), // MaterialPageRoute
      );
    } catch (e) {
      showDialog(
        context: context,
        builder: (context) => AlertDialog(
          shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
          backgroundColor: const Color(0xFFFFF3E0),
          title: const Row(
            children: [
              Icon(Icons.search_off, color: Colors.red, size: 28),
              SizedBox(width: 8),
              Text("Herb Not Found", style: TextStyle(color: Colors.red)),
            ],
          ),
        ),
      );
    }
  },
);
```

Figure 3.3.6. 3: A part code navigates to herb detail.

3.3.7 Notes Page Development

The Notes module enables users to create, edit, and manage personal notes within the application. It is composed of two main components: `NotesListPage` and `NoteEditorPage`.

The `NotesListPage` displays all saved notes retrieved from the `SharedPreferences` storage. Each note is uniquely identified by a timestamp-based ID and stored as a JSON object. Users can create new notes with default content, delete existing notes, or tap a note to open it for editing. The logic for loading, creating, and deleting notes is handled through dedicated functions and is illustrated in Figure 3.3.7.1.

When a note is opened, the `NoteEditorPage` loads the specific note based on its ID and populates the text fields with the existing content. Any changes to the title or content are automatically saved using the `_autoSave()` method, which updates the entire notes object in `SharedPreferences` without requiring manual confirmation. This auto-save behavior is shown in Figure 3.3.7.2.

All note data is stored locally using JSON encoding and decoding. This approach enables a lightweight and offline-capable record-keeping system for users to write and retain important personal information.

```

Future<void> _saveNotes() async {
  final prefs = await SharedPreferences.getInstance();
  await prefs.setString('notes', json.encode(_notes));
}

Future<void> _createNewNote() async {
  final newId = DateTime.now().millisecondsSinceEpoch.toString();
  _notes[newId] = {
    'title': 'Untitled Note',
    'content': '',
  };
  await _saveNotes();
  _loadNotes();
  Navigator.push(
    context,
    MaterialPageRoute(
      builder: (_) => NoteEditorPage(noteId: newId),
    ), // MaterialPageRoute
  );
}

Future<void> _deleteNote(String id) async {
  _notes.remove(id);
  await _saveNotes();
  _loadNotes();
}

```

Figure 3.3.7. 1: Code of note list page.

```

Future<void> _loadNote() async {
  final prefs = await SharedPreferences.getInstance();
  final raw = prefs.getString('notes');
  if (raw != null) {
    _allNotes = json.decode(raw);
    final note = _allNotes[widget.noteId];
    if (note != null) {
      _titleController.text = note['title'];
      _contentController.text = note['content'];
    }
  }
}

Future<void> _autoSave() async {
  setState(() => _isSaving = true);
  _allNotes[widget.noteId] = {
    'title': _titleController.text,
    'content': _contentController.text,
  };
  final prefs = await SharedPreferences.getInstance();
  await prefs.setString('notes', json.encode(_allNotes));
  Future.delayed(const Duration(seconds: 1), () {
    if (mounted) setState(() => _isSaving = false);
  }); // Future.delayed
}

```

Figure 3.3.7. 2: Code of note editor page.

3.3.8 Record Page

The Record module enables users to manage their saved herbs and formulas. This functionality is handled through the RecordPage component and two utility classes: FavoriteHerbStorage and FavoriteFormulaStorage.

Each time the app loads, the RecordPage uses the load() method from both storage classes to retrieve the user's saved data from SharedPreferences. These saved items are then categorized into Herbs and Formulas then displayed accordingly, as illustrated in Figure 3.3.8.1.

The saving and unsaving mechanism is managed through static methods toggleHerb() and toggleFormula(). When a user taps the bookmark icon in any herb or formula card, the corresponding object is either added to or removed from the local list, which is then encoded to JSON and saved back to SharedPreferences. This toggling logic is shown in Figure 3.3.8.2.

This module enables offline access to saved herbal data without requiring any backend integration. It provides users with a quick way to organize and return to frequently referenced herbs and formulas.

```
Future<void> _loadSavedItems() async {  
  await FavoriteHerbStorage.load();  
  await FavoriteFormulaStorage.load();  
  
  setState(() {  
    _savedHerbs = FavoriteHerbStorage.savedHerbs;  
    _savedFormulas = FavoriteFormulaStorage.savedFormulas;  
  });  
}
```

Figure 3.3.8. 1: Code of load both storage classes.

```

static Future<void> toggleFormula(Formula formula) async {
  final exists = savedFormulas.any((f) => f.formulaId == formula.formulaId);
  if (exists) {
    savedFormulas.removeWhere((f) => f.formulaId == formula.formulaId);
  } else {
    savedFormulas.add(formula);
  }
  await _save();
}

```

Figure 3.3.8. 2: Code of toggling logic.

3.3.9 Profile Page Development

The ProfilePage in this Flutter application provides a personalized user profile interface for authenticated users. Upon loading, it retrieves and displays the user's information in Figure 3.3.9.1, including their name and profile picture. The app distinguishes between Google-authenticated users and those registered via Firebase, dynamically fetching data either directly from Google credentials or the Firestore database. Additionally, it displays the app's current version using the `package_info_plus` package in Figure 3.3.9.2. The profile page features a centered user avatar, the user's name, and a logout button Figure 3.3.9.3 styled for emphasis. This streamlined design ensures a user-friendly experience while seamlessly integrating core functionalities like authentication and app metadata.

```
// from Firestore or Google load user data
Future<void> _loadUserData() async {
  User? user = _auth.currentUser;
  if (user != null) {
    // check is it use Google login?
    bool isGoogleSignIn = user.providerData.any((provider) => provider.providerId == "google.com");

    // if is use google login, use google's name
    if (isGoogleSignIn) {
      setState(() {
        userName = user.displayName ?? 'No Name';
      });
    } else {
      // or, from firestore get name
      DocumentSnapshot userDoc = await _firestore.collection('users').doc(user.uid).get();
      setState(() {
        userName = userDoc['name'] ?? 'No Name';
      });
    }
  }
}
```

Figure 3.3.9. 1: Load user data function.

```
// get app version
Future<void> _loadAppVersion() async {
  PackageInfo packageInfo = await PackageInfo.fromPlatform();
  setState(() {
    appVersion = packageInfo.version;
  });
}
```

Figure 3.3.9. 2: App version function.

```
// logout function
void _logout() async {
  await _auth.signOut();
  Navigator.pushReplacement(
    context,
    MaterialPageRoute(builder: (context) => LoginScreen()),
  );
}
```

Figure 3.3.9. 3: Log out function.

3.3.10 Herb Classification Page Development

The Herb Classification Page, implemented through CameraPage, enables users to scan and identify Chinese herbs using either real-time camera capture or gallery images. Two modes are available: Single Herb Mode and Multi Herb Mode, toggled via the bottom navigation bar as shown in Figure 3.3.9.1.

```
bottomNavigationBar: BottomNavigationBar(
  backgroundColor: Colors.black.withOpacity(0.6),
  selectedItemColor: Colors.green,
  unselectedItemColor: Colors.white,
  currentIndex: _currentIndex,
  onTap: (index) => setState(() {
    _currentIndex = index;
    _isMulti = index == 1;
  }),
  items: const [
    BottomNavigationBarItem(icon: Icon(Icons.looks_one), label: 'Scan Single'),
    BottomNavigationBarItem(icon: Icon(Icons.layers), label: 'Scan Multi'),
  ],
), // BottomNavigationBar
```

Figure 3.3.10. 1: Code of change to mode.

1. Camera Initialization & Image Acquisition

Upon page launch, the camera is initialized and permission is requested. The user may capture an image or upload one from the gallery. In Single Mode, the image is auto cropped to the center to improve classification accuracy (Figure 3.3.9.2).

```
Future<File> _cropCenter(String path) async {
  final bytes = await File(path).readAsBytes();
  final decoded = img.decodeImage(bytes)!;
  int w = decoded.width;
  int h = decoded.height;
  int size = w < h ? w : h;
  final cropped = img.copyCrop(decoded, x: (w - size) ~/ 2, y: (h - size) ~/ 2, width: size, height: size);
  final out = File('${path}_crop.jpg');
  await out.writeAsBytes(img.encodeJpg(cropped));
  return out;
}
```

Figure 3.3.10. 2: Code of crop center.

2. Model Inference via API

The herb classification process relies on model inference via Hugging face API. As shown in Figure 3.3.9.3 and Figure 3.3.9.4, when a user captures a photo using the camera or selects an image from the gallery, the image is sent to the appropriate API endpoint based on the selected scan mode: the EfficientNet API for single-herb classification, or the YOLO-based API for multi-herb detection. The server processes the image and returns prediction results in JSON format. These results are parsed on the client side to extract herb names. In single mode, the top-predicted herb is shown in a pop-up card, while in multi-mode, matched herbs are compared against the app's formula database to identify formulas with two or more ingredient matches. The matched formulas are presented to the user in an interactive pop-up interface.

```
final uri = Uri.parse(_isMulti ? multiAPI : efficientAPI);
final req = http.MultipartRequest('POST', uri);
req.files.add(await http.MultipartFile.fromPath('file', processed.path));
final res = await req.send();
final json = jsonDecode(await res.stream.bytesToString());
```

Figure 3.3.10. 3: Code to process API to recognise.

```
final String efficientAPI = "https://zhuwen0613-herb-model-api.hf.space/predict";
final String multiAPI = "https://zhuwen0613-herb-multidetector-api.hf.space/predict";
```

Figure 3.3.10. 4: Code to link the hugging face API.

3. Single Mode Result Display

In single mode, the top prediction is matched against the herb dataset. If found, a dialog displays the herb name and a button to view detail via HerbDetailPage (Figure 3.3.9.5). The dialog remains after returning from the detail page.

```

void _showResultPopUp(String label, double confidence) {
  final matchedList = _herbs.where(
    (herb) => herb.cnName == label || herb.engName.toLowerCase() == label.toLowerCase(),
  ).toList();
  final matched = matchedList.isNotEmpty ? matchedList.first : null;

  void showPopup() {
    showDialog(
      context: context,
      builder: (_) => Dialog(
        shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(24)),
        backgroundColor: const Color(0xFFFF6FFF8),
        child: Padding(
          padding: const EdgeInsets.all(20),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.min,
            children: [
              const Text(
                "🔍 Prediction Result",
                style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
              ), // Text
              const SizedBox(height: 16),
              InkWell(
                onTap: () async {
                  Navigator.of(context).pop();
                  if (matched != null) {
                    await Navigator.push(
                      context,
                      MaterialPageRoute(builder: (_) => HerbDetailPage(herb: matched)),
                    );
                  }
                },
              ),
            ],
          ),
        ),
      ),
    );
    showPopup();
  }
}

```

Figure 3.3.10. 5: Code of show single mode result.

4. Multi-Mode and Formula Matching

In multi-mode, detected herbs are matched against formulas. If ≥ 2 herbs in a formula are matched, the formula is considered a match and displayed in Figure 3.3.9.6. Tapping a card shows matched herb chips and allows navigation to the FormulaDetailPage (Figure 3.3.9.7).

```

.where((f) => (f['matchCount'] as int) >= 2)
.toList()
..sort((a, b) => (b['matchCount'] as int).compareTo(a['matchCount'] as int));

```

Figure 3.3.10. 6: Code to match the herb and the formula.

```

showDialog(
  context: context,
  barrierDismissible: false,
  builder: (BuildContext dialogContext) {
    return Dialog(
      backgroundColor: const Color(0xFFF6FFF8),
      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(24)),
      child: Padding(
        padding: const EdgeInsets.symmetric(vertical: 20, horizontal: 12),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.min,
          children: [
            const Text("🧪 Matched Formulas",
              style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold)), // Text
            const SizedBox(height: 12),
            ...matchedFormulas.map((f) => GestureDetector(
              onTap: () => _showFormulaDetailCard(f),
              child: Card(
                color: Colors.green[50],
                shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(20)),
                elevation: 5,
                margin: const EdgeInsets.symmetric(vertical: 8, horizontal: 12),
                child: Padding(
                  padding: const EdgeInsets.all(16),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Text("📌 ${f['name']}",
                        style: const TextStyle(fontSize: 18, fontWeight: FontWeight.bold)), // Text
                      const SizedBox(height: 6),
                      Text("Matched herbs: ${f['matchCount']} / ${f['herbs'] as List}.length}",
                        style: const TextStyle(color: Colors.black54)), // Text
                    ],
                  ),
                ),
            ),
          ],
        ),
      ),
    );
  },
);

```

Figure 3.3.10. 7: Code of navigate to formula details page.

5. Fallback Handling

If no formula is matched, a dialog explains the issue and still shows any individually recognized herbs for manual inspection (**Figure 3.3.9.8**).

```

if (matchedFormulas.isEmpty) {
  final hits = _herbs
    .where((h) => cnHits.contains(h.cnName))
    .toList();

  showDialog(
    context: context,
    builder: (_) => Dialog(
      backgroundColor: const Color(0xFFFD3F3F),
      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(24)),
      child: Padding(
        padding: const EdgeInsets.all(20),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.min,
          children: [
            // ✖ No Formula Matched
            Card(
              color: Colors.red[100],
              shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(16)),
              child: const Padding(
                padding: EdgeInsets.all(16),
                child: Column(
                  children: [
                    Text(
                      "✖ No Formula Matched",
                      style: TextStyle(
                        fontWeight: FontWeight.bold, fontSize: 18, color: Colors.red), // TextStyle
                    ), // Text
                  ],
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}

```

Figure 3.3.10. 8: Code to handle no formula matched.

6. Data Loading & Matching Logic

Herb and formula data are loaded from `all_herbs.json` and `herb_formulas.json` using `rootBundle.loadString()`. Predictions are matched by herb English name, then converted to Chinese names for formula matching (Figure 3.3.9.9).


```

Future<Void> _loadHerbs() async {
  final data = await rootBundle.loadString('assets/all_herbs.json');
  final List parsed = jsonDecode(data);
  setState(() => _herbs = parsed.map((e) => Herb.fromJson(e)).toList());
}

List<Formula> _formulas = [];

bool _isFormulaReady = false;

Future<Void> _loadFormulas() async {
  try {
    final data = await rootBundle.loadString('assets/herb_formulas.json');
    final Map<String, dynamic> parsed = jsonDecode(data);
    final List raw = parsed['formulas']; //

    setState(() {
      _formulas = raw.map((f) => Formula.fromJson(f)).toList();
      _isFormulaReady = true;
    });
  }
}

```

Figure 3.3.10. 9: Code to load the 2 JSON data.

3.3.10.1 Hugging Face API for Single Scan

To support remote herb image classification, a RESTful API was developed and deployed using Hugging Face Spaces. The backend was implemented using FastAPI and integrates a pre-trained TensorFlow Keras model. As shown in Figure 3.3.10.1.1, the application directory contains the main files including app.py, herb_model.keras, Dockerfile, and requirements.txt.

The model is loaded from the file herb_model.keras, which contains a trained EfficientNet-based classifier. Upon receiving an image via HTTP POST, the server resizes the image to 224x224 pixels, performs EfficientNet preprocessing, and makes a prediction. The top-3 classification results are returned in JSON format with confidence scores. The response format is demonstrated in Figure 3.3.10.1.2.

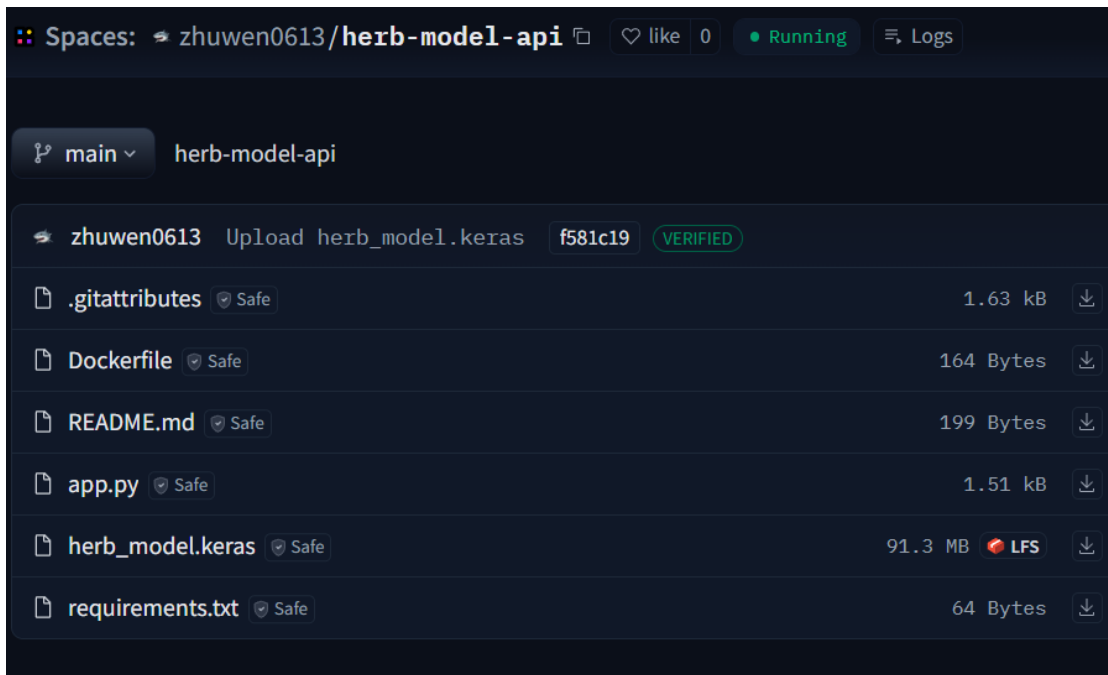


Figure 3.3.10.1. 1: Single scan hugging face.

```
def predict_image(image: Image.Image):
    image = image.resize((224, 224))
    img_array = np.array(image)
    img_array = tf.keras.applications.efficientnet.preprocess_input(img_array)
    img_array = np.expand_dims(img_array, axis=0)
    predictions = model.predict(img_array)[0]
    top3 = sorted(
        [{"label": labels[i], "confidence": float(predictions[i])} for i in range(len(predictions))],
        key=lambda x: x["confidence"],
        reverse=True
    )[:3]
    return top3
```

Figure 3.3.10.1. 2: Process of Efficient Net predict images.

3.3.10.2 Hugging Face API for Multi Scan

To support multi-herb detection in a single image, another RESTful API was developed using FastAPI and the Ultralytics YOLOv8 framework. The API was deployed on Hugging Face Spaces and handles object detection with bounding box output. As shown in **Figure 3.3.10.2.1**, the repository includes core files such as app.py, best.pt, requirements.txt, and Dockerfile.

The detection model is loaded from best.pt, a YOLOv8 PyTorch checkpoint trained on a custom herb dataset. Upon receiving an image via the /predict endpoint, the API processes the input with YOLOv8 and extracts the bounding boxes, class indices, and

confidence values. The prediction results are then formatted into a list of herb labels and corresponding confidence scores, as shown in **Figure 3.3.10.2.2**.

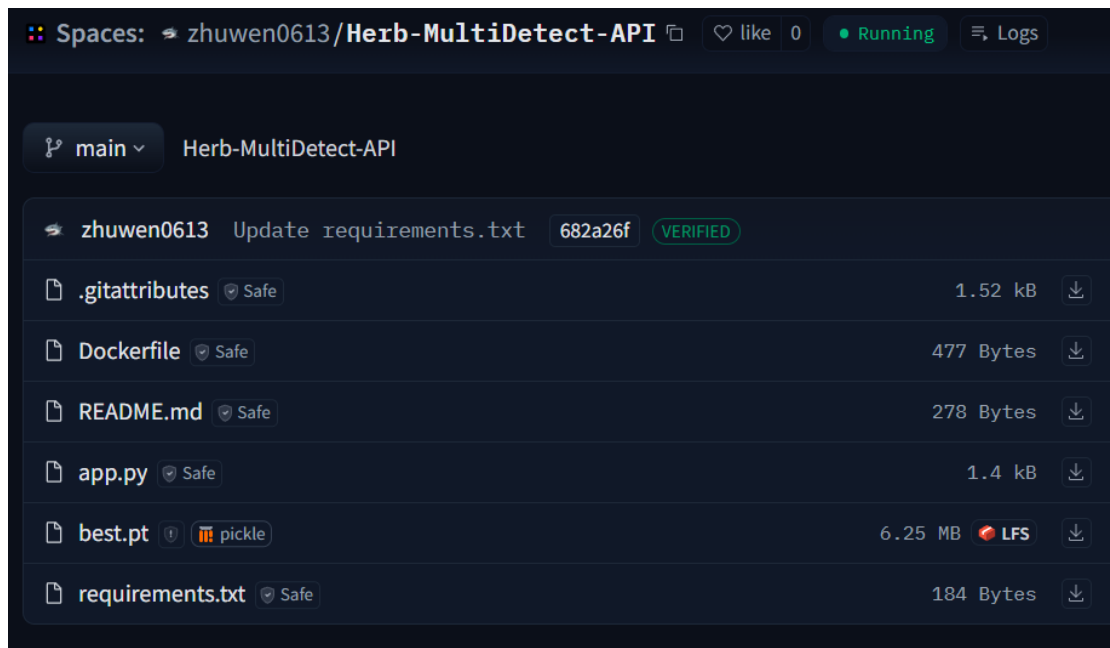


Figure 3.3.10.2. 1: Multi scan hugging face.

```
@app.get("/")
async def root():
    return {"message": "Herb MultiDetect API is running."}

@app.post("/predict")
async def predict(file: UploadFile = File(...)):
    try:
        # Read the uploaded Image
        image = Image.open(io.BytesIO(await file.read())).convert("RGB")
    except Exception:
        return JsonResponse(content={"error": "Invalid image file"}, status_code=400)

    # Model Predict
    results = model.predict(image, imgsz=640)
    detections = results[0].boxes.data.tolist()

    # output format
    output = []
    for box in detections:
        x1, y1, x2, y2, conf, cls = box
        output.append({
            "label": class_names[int(cls)],
            "confidence": round(float(conf), 3),
            "box": [round(x1), round(y1), round(x2), round(y2)]
        })

    return JsonResponse(content={"predictions": output})
```

Figure 3.3.10.2. 2: Process of YOLO predict images.

3.3.11 Model Architecture

3.3.11.1 Efficient Net Model

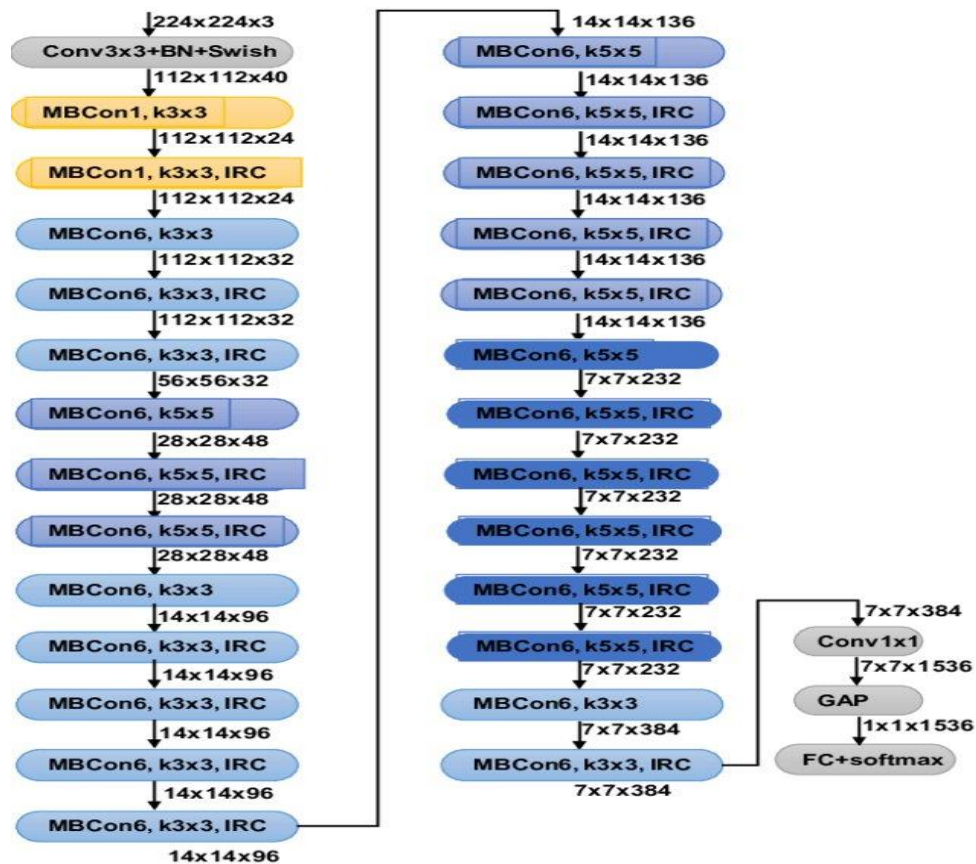


Figure 3.3.11.1. 1: EfficientNet-B3 Architecture.

EfficientNet B3 is a convolutional neural network architecture known for its high efficiency in balancing model size and accuracy. It uses compound scaling to optimize depth, width, and resolution, achieving better performance with fewer computational resources. The network consists of several layers, including mobile inverted bottleneck convolution (MBConv) blocks, which enhance feature extraction while reducing computational complexity. It also incorporates features like Swish activation and squeeze-and-excitation blocks to improve learning capability.

For this project, the EfficientNet B3 pre-trained model was utilized, which allows leveraging its pre-learned weights and architecture to fine-tune on herb classification tasks. After training the model on the specific herb dataset, the architecture produced a `herb_model.keras` file containing the trained model weights and configuration.

3.3.11.2 YOLO Architecture

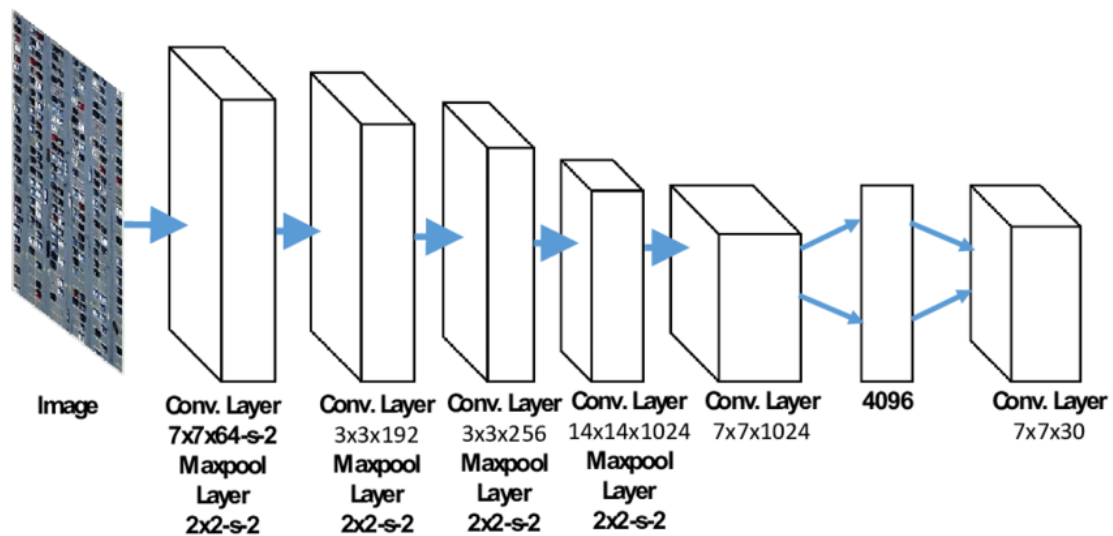


Figure 3.3.11.2. 1: YOLO Architecture.

YOLOv8 (You Only Look Once version 8) is a real-time object detection architecture known for its speed and accuracy. It is based on a single-stage detection pipeline, where the input image is processed through a series of convolutional layers to simultaneously predict bounding boxes and class labels. The network performs detection in one unified pass, making it highly suitable for fast herb identification in mobile applications.

YOLOv8 includes improvements over previous versions such as decoupled head, anchor-free prediction, and adaptive training techniques. It supports custom training on small datasets and has native export support for ONNX, TFLite, and other formats.

For this project, YOLOv8 was adopted for multi-herb detection. The model was trained on a custom herb image dataset annotated with bounding boxes. After training, the model was exported to best.pt, which was then deployed to Hugging Face and accessed via API. This enabled the mobile app to upload an image and receive multiple detected herb results with location and confidence scores.

3.4 Summary

This chapter presented the complete system design and implementation details of the Herb Scanner App. Beginning with the use case diagram, it outlined all functional interactions between the user and the system, including core features such as herb scanning, record management, note taking, and user authentication.

Subsequent sections elaborated on the technical realization of each module. The herb classification system supports both single and multi-herb recognition modes, powered by EfficientNet and YOLOv8 models respectively. These models were deployed to Hugging Face and integrated into the app through API endpoints. Each recognition result is matched against a local herb and formula dataset, enabling accurate identification and intelligent formula suggestions.

In addition to recognition, features like profile viewing, note writing, herb browsing, and saving to record were implemented to support a full user experience. Data management through SharedPreferences, structured UI handling, and API integration were all tailored for performance, clarity, and offline usability.

In summary, these components demonstrate a robust and well-integrated mobile solution for traditional Chinese herb recognition and information access.

CHAPTER 4 – Methodology and Tools

4.1 Methodologies

The methodology proposed for the project is the Rapid Application Development (RAD) methodology. RAD is an adaptive software development methodology that emphasizes quick and iterative development cycles, making it an ideal approach for projects requiring flexibility and responsiveness to change. By focusing on prototyping and user feedback, RAD allows for continuous refinement and enhancement of the application throughout the development process. This methodology is particularly suitable for projects that involve integrating complex features like image recognition and personalized recommendations, as it enables developers to rapidly build, test, and improve these components based on real-time insights and evolving requirements. With its emphasis on component reusability and collaborative development, RAD supports the efficient creation of a functional and user-friendly mobile application, ensuring that the final product meets the users' needs effectively.

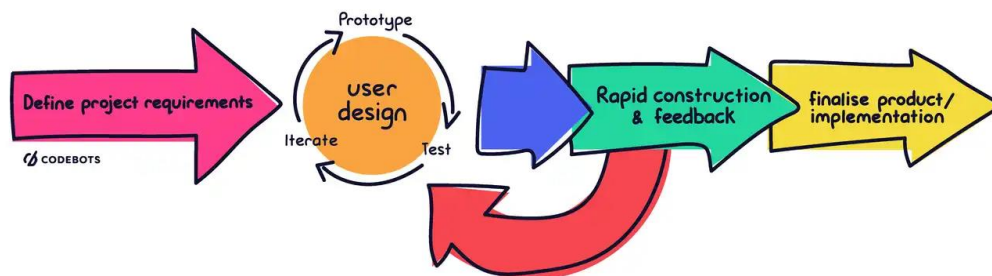


Figure 4.1. 1: Process of RAD [5].

4.1.1 Requirements Planning Phase

At the beginning of the project, the objectives were clearly defined by analyzing existing traditional Chinese medicine applications and identifying their limitations. The goal was to develop a mobile application capable of recognizing Chinese herbs from images and recommending relevant formulas based on recognized herbs. Literature and

application reviews were conducted to guide the design decisions, especially concerning herb recognition accuracy and system usability. Android was selected as the deployment platform. TensorFlow was used to train the image classification model, and YOLOv8 was later introduced for multi-object detection. The application was planned to run fully on mobile devices, with some cloud-based support via Hugging Face APIs.

4.1.2 User Design

During the User Design Phase, the overall structure and visual flow of the application were planned in detail based on the requirements established earlier. Key user-facing modules were identified, including user login and registration, real-time camera integration, image-based herb classification, multi-herb formula matching, and the record module for storing favorited items. For each module, interaction flows were carefully drafted to ensure a smooth and intuitive user experience. The database design was structured using local JSON files that store herb and formula information. These files were selected over online databases to ensure offline access, quick loading, and minimal latency when matching recognized herbs to internal data. The application uses the Flutter framework to develop cross-platform user interfaces with high rendering performance. Each screen was designed with consistency and logical transitions in mind, especially between camera scanning, result dialogs, and detail navigation. SharedPreferences was chosen for its simplicity and efficiency in handling local persistent storage of saved herb and formula data.

4.1.3 Construction and Feedback Phase

The Construction and Feedback Phase focused on the actual development and iterative refinement of all core modules. The first components developed included the camera module and the single-herb recognition system. Initially, MobileNet was adopted as the image classification model due to its lightweight nature and mobile compatibility. However, after initial testing, its limited feature extraction capability led to unsatisfactory accuracy levels. As a result, the model was replaced with EfficientNet B3, which significantly improved performance while maintaining efficiency.

The application was then expanded to support multi-herb detection using a custom-trained YOLOv8 model. This model was trained using annotated herb datasets and exported as a PyTorch weight file. Both the EfficientNet and YOLO models were deployed to Hugging Face Spaces and made accessible through public API endpoints. The application's backend API logic was developed using FastAPI, allowing seamless POST requests from the Flutter frontend.

Recognition results from both models were parsed and matched with local herb data. In multi-herb mode, matched herbs were used to query formulas, and those with two or more herb overlaps were displayed in sorted popups. Additional logic was implemented to handle edge cases, such as no herbs detected, or formulas with unmatched components.

Throughout this phase, the system was tested repeatedly. Feedback was gathered regarding recognition speed, result accuracy, and usability. Based on this input, various optimizations were made, including preloading local data, improving loading states, and refining error dialogs. This phase ensured the core logic of the application was stable and responsive across typical usage scenarios.

4.1.4 Finalize Product and Implementation Phase

In the Finalize Product and Implementation Phase, all system components were consolidated and subjected to comprehensive testing. Each functional module was validated to ensure it met project requirements and delivered consistent outputs. The user interface was fine-tuned to enhance clarity and reduce interaction steps. Scan button response timing, popup animations, and page transitions were carefully adjusted to maintain a responsive user experience. SharedPreferences logic was tested to verify that bookmarked items persisted across sessions and could be removed or updated without error. System-wide testing included camera permission handling, image upload timing, model response delays, and multi-step navigation flow between result screens and detail views. The app was deployed and tested on real Android devices to verify platform-specific behavior and performance.

After validating all features, final refinements were made based on feedback from testers. These adjustments included improving the fallback behavior when no herbs or formulas were detected, enhancing pop-up readability, and revising the bottom navigation logic for mode switching. Once stability and functionality were confirmed, the application was considered ready for use in real-world scenarios.

4.2 Tools to Use

4.2.1 Hardware

The hardware involved in this project is computer and mobile device. The computer is used for fine-tuning the model, while the mobile device integrates all the functionalities and serves as the platform for testing and scanning TCM herbs. There are listed down in Table 4.2.1.1 and Table 4.2.1.2.

Table 4.2.1. 1: Specifications of laptop.

Description	Specifications
Model	Lenovo IdeaPad Gaming 3
Processor	AMD Ryzen 5 5600H with Radeon Graphics
Operating System	Windows 11
Graphic	1. AMD Radeon™ Graphics 2. NVIDIA GeForce GTX 1650
Memory	16GB DDR4 RAM
Storage	1.5TB SSD

Table 4.2.1. 2: Specifications of mobile device.

Description	Specifications
Model	Xiaomi Poco X6 Pro

Processor	Mediatek Dimensity 8300 Ultra
Basic frequency	3.35 GHZ
Operating System	Android 14, HyperOS
Memory	8GB
Storage	256GB
Screen resolution	1120 x 2712
Camera	64 MP

4.2.2 Software

Table 4.2.2. 1: Specifications of software.

Description	Specifications
Tools	<p><u>Android Studio</u></p> <p>An integrated development environment (IDE) for Android development. It provides tools for designing, coding, testing, and debugging Android applications. It's the primary tool for compiling, testing, and deploying your Flutter app on Android devices.</p>
	<p><u>Firebase</u></p> <p>A platform developed by Google for creating mobile and web applications. It offers a variety of services like real-time databases, authentication, analytics, and cloud storage. Firebase is can used for storing and syncing data across mobile app, such as user profiles and history.</p>
	<p><u>Colab</u></p> <p>Google Colab (Colaboratory) is a free cloud-based platform that provides Jupyter notebook environments. It allows users to write and execute Python code in the browser, making it easy to work with machine learning, data analysis, and other computing tasks. Colab offers free access to GPUs and TPUs, which significantly speeds up</p>

	<p>model training and testing. It's widely used for developing and experimenting with deep learning models, making it an ideal tool for training models before deploying them to mobile applications.</p>
	<p><u>Hugging Face</u></p> <p>Hugging Face is a platform for hosting, sharing, and deploying machine learning models via APIs. In this project, Hugging Face is used to deploy both the EfficientNet-based single-herb recognition model and the YOLOv8 multi-herb detection model. These models are accessible through cloud APIs, enabling the mobile application to perform real-time herb identification by sending images to the hosted endpoints.</p>
Languages	<p><u>Dart</u></p> <p>The primary programming language used in Flutter for building cross-platform mobile apps. It is optimized for UI, offering features like a strong type of system, sound null safety, and async-await syntax for managing asynchronous operations.</p>
	<p><u>Python</u></p> <p>Python is a versatile, high-level programming language known for its simplicity and readability. It is widely used in data science, machine learning, and web development. In the context of app development, Python is often used for backend services, AI integration, and deep learning tasks. Python can be employed for training machine learning models, especially using frameworks like TensorFlow or PyTorch, and later integrating those models into mobile apps.</p>
Libraries and Frameworks	<p><u>Flutter</u></p> <p>A UI toolkit from Google for building natively compiled applications for mobile, web, and desktop from a single codebase. It allows for fast development, expressive UIs, and native performance.</p>

	<p><u>SharedPreferences</u></p> <p>A lightweight data storage solution in Android that allows to store small amounts of data in key-value pairs. This is often used for storing user preferences, configuration settings, or other small amounts of data that need to be persisted across user sessions.</p>
	<p><u>YOLO (You Only Look Once)</u></p> <p>YOLO is a state-of-the-art, real-time object detection algorithm widely used in computer vision tasks. It can detect multiple objects in a single image quickly and accurately. In this project, YOLO is utilized for multi-herb detection, enabling the application to identify several herbs simultaneously from one photo. The model is trained using YOLOv8 and deployed via a cloud API for real-time inference.</p>

4.3 Gantt Chart



Figure 4.3. 1: Gantt Chart 1.

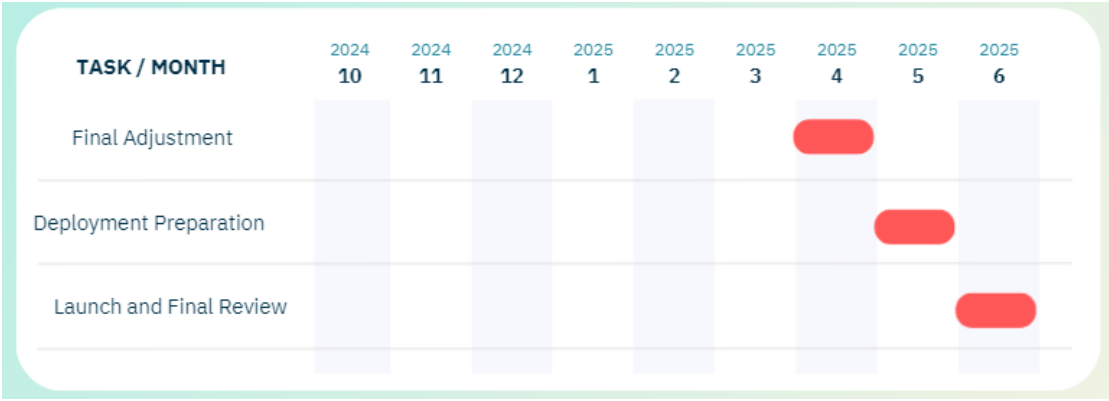


Figure 4.3. 2: Gantt Chart 2.

4.4 Summary

This chapter summarized the methodologies, development resources, and technologies utilized throughout the creation of the Herb Scanner App. Adopting the Rapid Application Development approach, the project progressed through clearly defined stages, including Requirements Planning, User Design, Construction and Feedback, and Finalization. Detailed software specifications such as Android Studio, Flutter, Firebase, TensorFlow-based EfficientNet B3, YOLOv8, and Hugging Face APIs were employed to achieve accurate herb recognition and recommendation functionalities. Comprehensive hardware and software testing ensured that the application performed reliably on Android devices. User feedback was actively incorporated to enhance the application's usability, performance, and user experience. As a result, the finalized app successfully fulfilled the targeted objectives, providing users with an effective, accurate, and user-friendly tool for traditional Chinese medicine herb identification and recommendation.

CHAPTER 5 – Implementation and Testing

All 10 main pages of the Herb Scanner App were successfully developed, including the login page, register page, view type of herb page, view type of formula page, herb detail page, formula detail page, scan classification page, record page, notes page, and profile page. The application is designed with a clean and minimal interface using Flutter, and it is fully compatible with Android devices. Most pages include consistent navigation patterns such as bottom navigation and structured card layouts for data display.

The login module supports both email-password and Google authentication, while the register module allows new users to sign up with email credentials. The herb and formula viewing modules allow users to browse available TCM data stored in local JSON files. Users can scan herbs through camera or gallery, with support for both single and multi-herb recognition. Recognized results are matched to formula data and displayed in interactive popups. Bookmarked herbs and formulas are stored locally using SharedPreferences and displayed in the record page. Notes can be created and saved by users, and profile information including name and avatar is shown on the profile page. All functionalities were tested for stability, accuracy, and responsiveness across supported devices.

5.1 Login Page

When a user downloads and launches the app, they are directed to the login page in Figure 5.1.1. New users can register an account by clicking the provided registration link and signing up with their email and password. Alternatively, users have the option to sign in directly using their Google account, providing a seamless and convenient authentication process. This dual-option login system ensures flexibility and accessibility for a broad range of users.

Upon entering an email and password, the system verifies the credentials. If the login is successful, the user is redirected to the herb page, granting access to the app's main features. After a successful login, the application uses SharedPreferences to save the user's login state, enabling automatic login in future sessions unless the user logs out

manually. If the login fails, an alert notifies the user that the email or password entered is incorrect.

Additionally, for users who forget their passwords, a "Forgot Password?" link is provided. Clicking this link prompts the system to send a password reset email to the user's registered address, allowing them to regain access to their account efficiently.

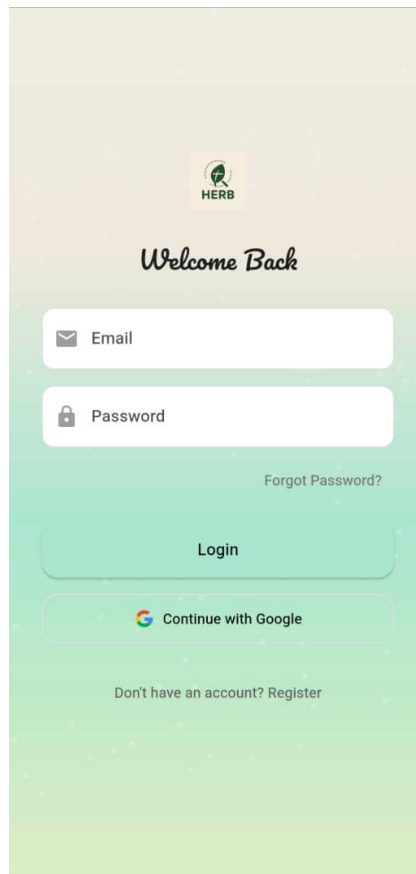


Figure 5.1. 1: Login Page.

5.2 Register Page

On the register page in Figure 5.2.1, users are required to provide their name, a valid email address, and a password with a minimum of six characters. Once the registration is successful, the user's name and email are stored in Firestore under a dedicated users collection for future access and personalization. The user is then automatically redirected to the login page, allowing them to sign in and begin using the application.

If the registration fails, such as when the email address is already in use, the system displays an error message to inform the user. This ensures a smooth and informative registration process while maintaining data integrity.

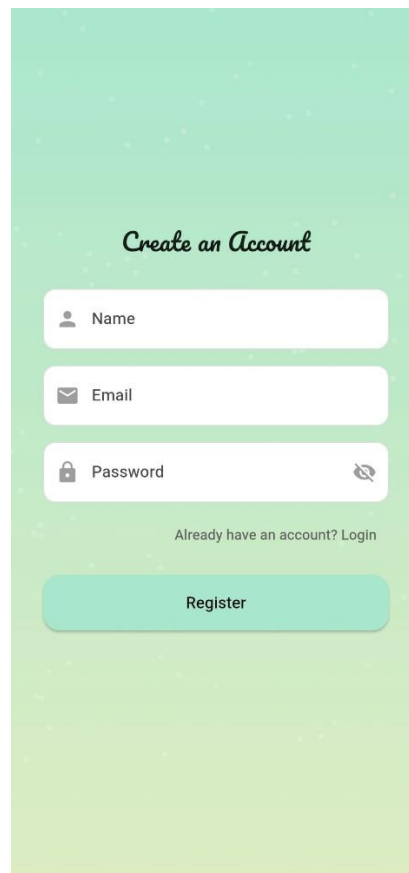
The image shows a mobile application interface for creating an account. The background is a light green gradient with small white dots. At the top, the text "Create an Account" is centered in a black, cursive-style font. Below this, there are three white input fields with rounded corners. The first field is labeled "Name" with a person icon on the left. The second field is labeled "Email" with an envelope icon on the left. The third field is labeled "Password" with a lock icon on the left and an eye icon on the right to toggle visibility. Below the input fields, the text "Already have an account? Login" is centered in a small, grey font. At the bottom, there is a large, rounded green button with the word "Register" in white text.

Figure 5.2. 1: Register Page.

5.3 Herb Type Page

After successfully logging in, users are directed to the herb type page in Figure 5.3.1, where they can explore all available herbs. Each herb is presented in a card format, displaying a photo of the herb, its Chinese and English names, and its property label, such as warm or cold.

Users can search for herbs by entering their Chinese or English names into the search bar. Additionally, the page supports property-based filtering in Figure 5.3.2, allowing users to select options such as warm, cold, hot, or neutral to refine the results. The layout provides an intuitive and efficient browsing experience, enabling users to quickly locate and learn about traditional herbs.

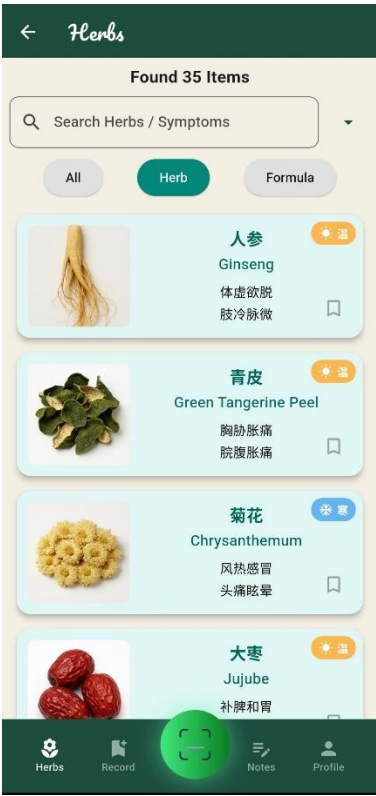


Figure 5.3. 1: Herb Type Page.

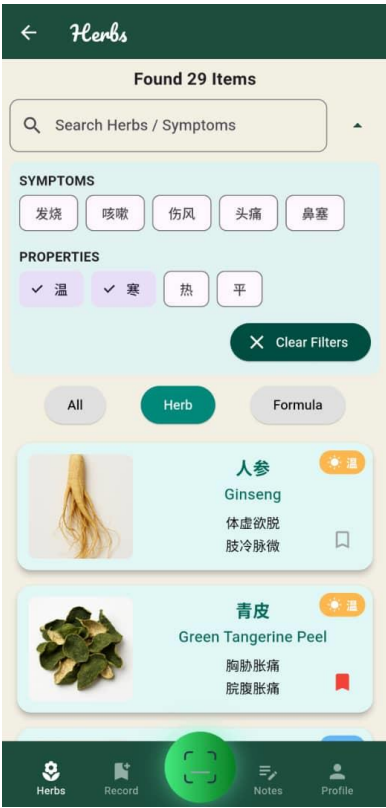


Figure 5.3. 2: Filter of herb's properties.

5.4 Herb Detail Page

When users want to learn more about a specific herb, they can tap the corresponding card on the herb type page to access the herb detail page, as shown in Figure 5.4.1. This page provides detailed information about the selected herb, including a high-resolution image, the herb's name in Chinese and English, and its Latin name if available.

The page displays multiple sections such as description, functions, indications, medicinal properties, usage part, recommended dosage, contraindications, and associated meridian channels. All relevant content is organized into clearly labeled cards to enhance readability. For data cleanliness, any section with missing or undefined values is automatically excluded from display.

The indication section provides insight into the herb's traditional clinical applications, helping users understand what symptoms or conditions the herb is commonly used to address. Overall, the herb detail page serves as a complete reference hub, allowing users to explore the full profile of each herb in a well-structured and informative format.



Figure 5.4. 1: Herb Detail Page.

5.5 Formula Type Page

The formula type page, as shown in Figure 5.5.1, allows users to explore a comprehensive list of traditional Chinese medicine formulas. Each formula is displayed in a card format, presenting an image of the combined herbs, the formula's name in both Chinese and English, and its related indications. This organized layout allows users to conveniently browse and identify different herbal combinations used in TCM practices.

A search bar is available at the top of the page, supporting searches by both the Chinese and English names of formulas. In addition, users can search based on symptoms in Figure 5.5.2, making it easier to locate formulas that are relevant to specific health conditions. To further enhance usability, a quick filter section enables users to filter formulas by selected symptoms. The filtering system dynamically updates the list of displayed formulas, ensuring only relevant results are shown. This page serves as a centralized hub for discovering and accessing detailed information about various herbal prescriptions.



Figure 5.5. 1: Formula type page.



Figure 5.5. 2: Filter of symptoms.

5.6 Formula Detail Page

When users tap on a formula card from the formula type page, they are navigated to the formula detail page, as shown in Figure 5.6.1. This page provides detailed information about the selected herbal formula, including a high-resolution image, the formula's name in both Chinese and English, and a brief description of its traditional use.

Additional sections include the formula's indications, which outline the symptoms or conditions the formula is typically used to treat, and a comprehensive list of herbs that make up the formula. Each herb is grouped according to its functional role in the formula, such as Chief, Deputy, Assistant, or Envoy. The user can tap on any herb listed in the composition to access its corresponding detail page for further information. In the event a listed herb does not exist in the database, an error dialog will notify the user accordingly.

All available information is organized into color-coded sections for clarity and consistency. The formula detail page serves as an educational and reference tool for users to understand the composition and applications of traditional Chinese medicine prescriptions.



Figure 5.6. 1: Formula detail page.

5.7 Record Page

The record page, as shown in Figure 5.7.1, allows users to view all the herbs and formulas in Figure 5.7.2 they have bookmarked. The interface is divided into two tabs: "Herbs" and "Formulas." Each saved item is displayed in the same card layout used throughout the app, providing familiar and consistent user experience. Users can quickly browse through their saved items for easy reference without having to search for them again in the main herb or formula listing. Each item includes a bookmark icon, which users can tap to remove the item from their saved list. This feature helps users manage their personalized herbal reference list efficiently and ensures that only relevant and useful items are retained.



Figure 5.7. 1: Record page in herbs.



Figure 5.7. 2: Record page in formulas.

5.8 Notes Page

The Notes Page, as illustrated in Figure 5.8.1 and Figure 5.8.2, provides users with a personal space to jot down and organize their own health-related notes. Upon accessing the page, users are presented with a list of previously saved notes. Each note is displayed with its title, and users can tap on any note to view or edit its content. A delete icon is also provided next to each note, allowing for quick removal when necessary.

Users can create new notes by clicking the "+" button located at the bottom right corner. Each note is assigned a unique ID and stored locally using the SharedPreferences mechanism, ensuring offline accessibility and persistence. As users type, the content is automatically saved, indicated by a subtle "Auto Saving..." message at the top of the editor screen. This feature ensures that user input is never lost, even if the app is accidentally closed or interrupted.

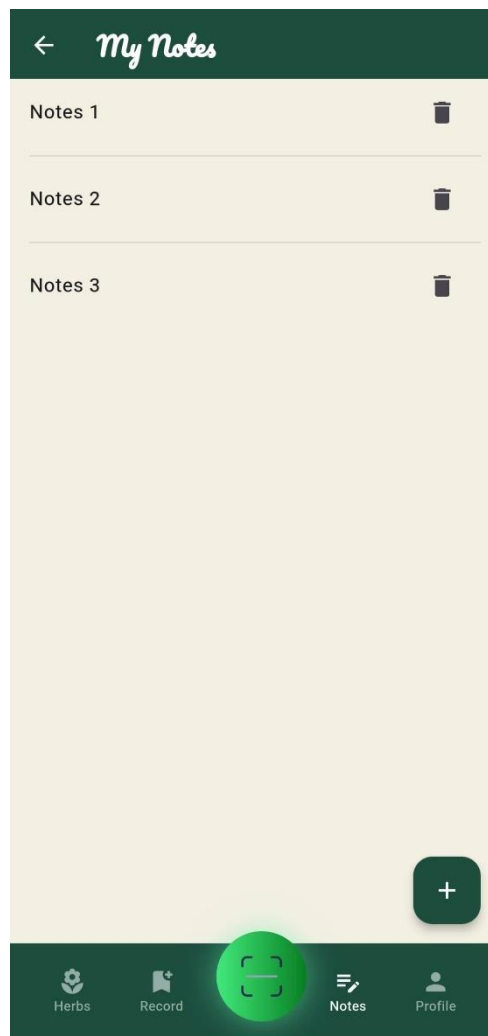


Figure 5.8. 1: Notes page 1.

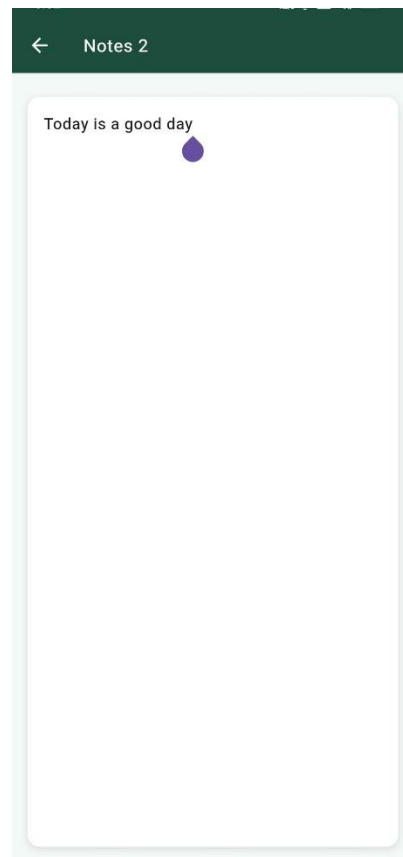


Figure 5.8. 2: Notes page 2.

5.9 Profile Page

The profile page, as illustrated in Figure 5.9.1, allows users to conveniently view their account information, including their display name, profile picture, and the current version of the app. The layout is minimal and clean, aligning with the overall aesthetic of the app. If the user has signed in using a Google account, their Google profile image will be displayed. For users registered via email, no profile image is shown, providing a simple visual distinction between authentication methods. While profile editing features are not currently available, the page includes a prominent “Logout” button. Tapping this button securely signs the user out of their session, returning them to the login screen. This section ensures that users can confirm their login status and app version while also offering a quick way to log out, supporting session control and account privacy.

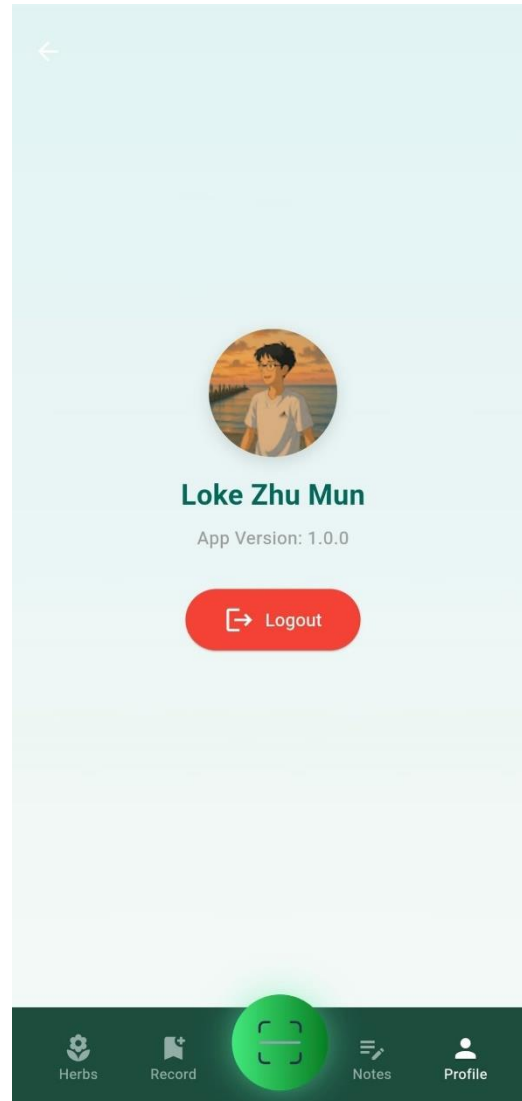


Figure 5.9. 1: Profile Page.

5.10 Herb Classification Page

The Herb Classification Page allows users to classify herbs using their mobile device's camera or gallery images. As shown in Figure 5.10.1, users can choose between two modes at the bottom of the screen: Scan Single for identifying one herb at a time, or Scan Multi for detecting multiple herbs in a single image. The UI also includes a translucent scan frame and flashlight toggle to assist in image clarity.

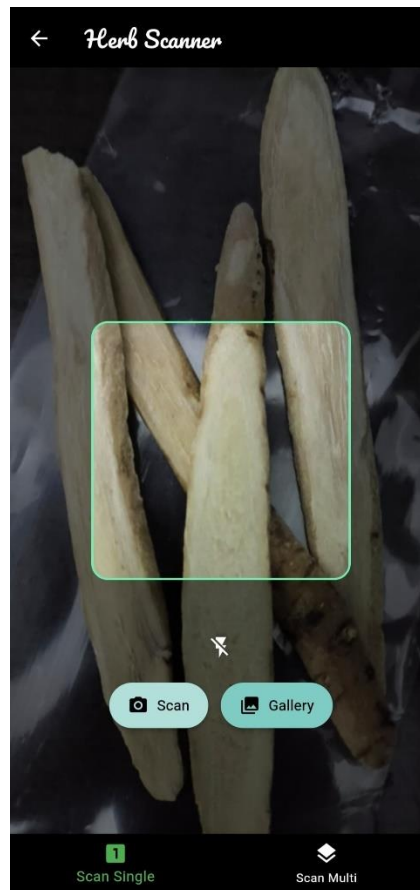


Figure 5.10. 1: Single scan.

When the Scan Single mode is selected, users point their camera at a herb and press the "Scan" button or select an image from the gallery. The captured image is then sent to the EfficientNet-based classification API for inference. If a herb is successfully detected, a pop-up dialog (Figure 5.10.2) displays the predicted herb name and a shortcut to view detailed information. This offers a quick and accurate way to identify herbs based on visual input.

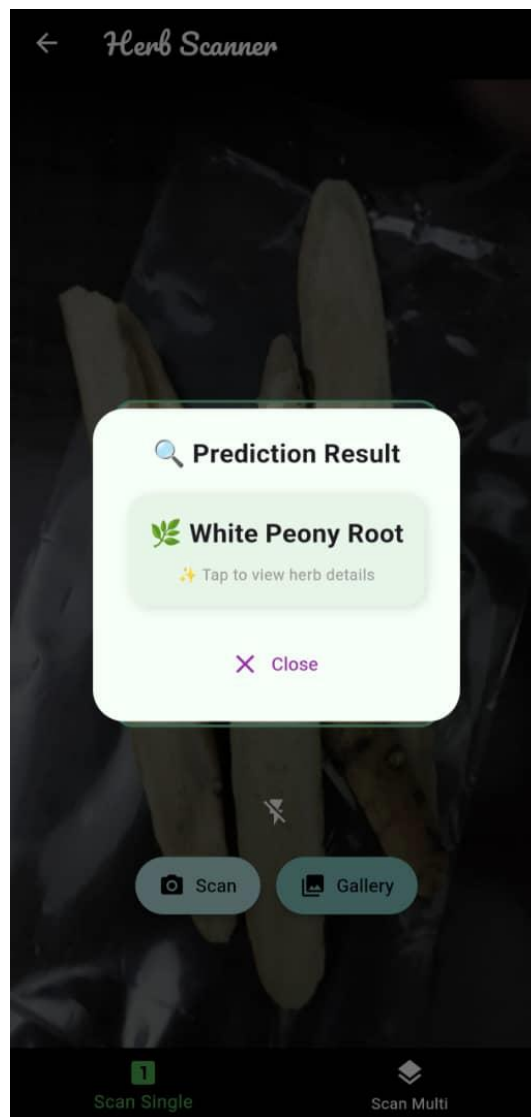


Figure 5.10. 2: Single scan successfully.

For more complex cases, the Scan Multi mode can be selected (Figure 5.10.3), allowing the system to recognize multiple herbs in one photo using the YOLOv8 detection model. Upon detection, the system checks how many herbs match each formula from the internal database. If a formula contains at least two matching herbs, it is shown in a Matched Formulas pop-up (Figure 5.10.4). Users can then tap on any matched formula card to view the detailed ingredient match breakdown in Figure 5.10.5, where matched herbs are highlighted.



Figure 5.10. 3: Multi scan.



Figure 5.10. 4: Multi scan successfully.

In the event that no formula matches the detected herbs, the system displays a fallback dialog (Figure 5.10.6) showing which individual herbs were recognized. Users can still tap these herbs to view their detail pages, providing helpful information even when no complete formula is matched.

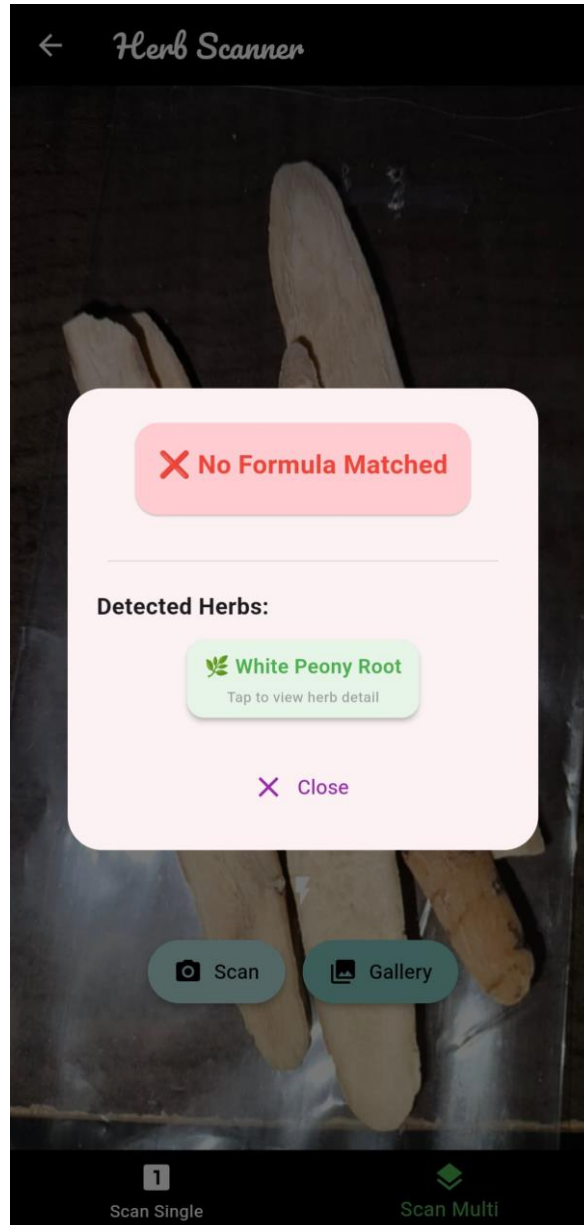


Figure 5.10. 5: Multi scan no match formula.

5.11 TensorFlow Model

5.11.1 Dataset

The original herb image dataset used for this project was sourced from articles [3], consisting of 95 classes and a total of 5,640 images. However, to improve model accuracy and ensure better quality control, the dataset was manually cleaned and reorganized, focusing only on 13 commonly used herbs. Images were filtered based on quality, relevance, and consistency, resulting in a final working dataset of 13 classes and 941 images, as illustrated in Figure 5.11.1.1.

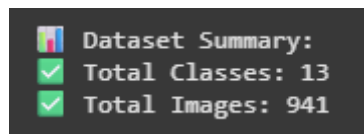


Figure 5.11.1. 1: Total dataset

5.11.2 Splitting the Dataset

This dataset is divided into 70% training set (659 images) and 30% testing set (282 images), as illustrated in Figure 5.11.2.1 and Figure 5.11.2.2. The splitting process is handled by a custom function, which ensures that each class is proportionally represented in both the training and testing sets.

The decision to allocate 30% to the testing set stems from the uneven distribution of image samples across different herb classes. If an additional validation set were to be created, some classes might contain too few samples, reducing training effectiveness and harming generalization. Therefore, in this project, the testing set is also used as the validation set, striking a balance between model monitoring (overfitting or underfitting) and retaining enough data for reliable evaluation across all 13 classes.

```

# Evenly split the dataset into training and testing sets
def split_data_equally(data, test_size=0.3):
    train_data = []
    train_labels = []
    test_data = []
    test_labels = []
    for class_name, images in data.items():
        if len(images) == 0:
            print(f"Skipping empty class: {class_name}")
            continue
        train_images, test_images = train_test_split(images, test_size=test_size, random_state=42)
        print(f"Class: {class_name}, Train: {len(train_images)}, Test: {len(test_images)}")
        train_data.extend(train_images)
        train_labels.extend([class_name] * len(train_images))
        test_data.extend(test_images)
        test_labels.extend([class_name] * len(test_images))
    return train_data, test_data, train_labels, test_labels

```

Figure 5.11.2. 1: Split data to 70% training set and 30% testing set.

```

Class: Angelica, Train: 79, Test: 35
Class: Jujube, Train: 46, Test: 21
Class: Frankincense, Train: 44, Test: 19
Class: Chrysanthemum, Train: 37, Test: 17
Class: Prepared Rehmannia Root, Train: 45, Test: 20
Class: Schisandra, Train: 35, Test: 15
Class: Rhubarb, Train: 61, Test: 27
Class: Hawthorn Fruit, Train: 42, Test: 18
Class: Turtle Carapace, Train: 35, Test: 15
Class: White Peony Root, Train: 58, Test: 26
Class: Chuanxiong Rhizome, Train: 48, Test: 21
Class: Lotus Seed, Train: 81, Test: 35
Class: Ginseng, Train: 42, Test: 19

```

Figure 5.11.2. 2: Distribution of dataset after splitting.

5.11.3 Data Augmentation

To improve the model's robustness and generalization ability, data augmentation was applied to the training dataset using a range of image transformation techniques, as shown in Figure 5.11.3.1. These transformations include:

- Random rotations up to 45°
- Width and height shift up to 30%
- Shear transformations up to 30%
- Zooming within a range of 40%
- Brightness adjustments between 70% and 130%
- Horizontal and vertical flipping

All images were resized to 224×224 pixels, which is the input size expected by EfficientNet. Additionally, EfficientNet’s built-in preprocess_input function was used to normalize the image data. These augmentations significantly increase the diversity of the training set without requiring more actual images, reducing the risk of overfitting. Some augmented image samples are shown in Figure 5.11.3.2.

```
def create_generators(train_data, train_labels, test_data, test_labels, img_size=(224, 224), batch_size=16):
    train_datagen = ImageDataGenerator(
        preprocessing_function=tf.keras.applications.efficientnet.preprocess_input,
        rotation_range=45,
        width_shift_range=0.3,
        height_shift_range=0.3,
        shear_range=0.3,
        zoom_range=0.4,
        brightness_range=[0.7, 1.3],
        horizontal_flip=True,
        vertical_flip=True,
        fill_mode='nearest'
    )
    test_datagen = ImageDataGenerator(preprocessing_function=tf.keras.applications.efficientnet.preprocess_input)
```

Figure 5.11.3. 1: Augmentation and resize code.



Figure 5.11.3. 2: Sample of augmentation and resize.

5.11.4 Model Architecture

The model leverages EfficientNetB3 in Figure 5.11.4.1, a state-of-the-art pre-trained architecture known for its efficiency and accuracy in image classification tasks. The base EfficientNetB3 layers serve as a feature extractor, with around 60% of its layers frozen to preserve the general feature representations learned from the ImageNet dataset, ensuring efficient transfer learning. A GlobalAveragePooling2D layer follows, reducing the spatial dimensions of the feature maps while retaining essential information. The extracted features are passed through a fully connected dense layer with 256 units and ReLU activation, promoting non-linearity for better classification performance. To prevent overfitting, a Dropout layer is incorporated before the final output layer, which uses softmax activation to provide probabilities for each of the 95 classes in the dataset. The model is compiled with the Adam optimizer and categorical cross-entropy loss for effective optimization during training.

Model Summary:
Model: "sequential_3"

Layer (type)	Output Shape	Param #
efficientnetb3 (Functional)	(None, 7, 7, 1536)	10,783,535
global_average_pooling2d_3 (GlobalAveragePooling2D)	(None, 1536)	0
dense_6 (Dense)	(None, 256)	393,472
dropout_3 (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 95)	24,415

Total params: 11,201,422 (42.73 MB)
Trainable params: 5,671,341 (21.63 MB)
Non-trainable params: 5,530,081 (21.10 MB)

Figure 5.11.4. 1: EfficientnetB3 Model Architecture.

5.11.5 Model Training

The training process begins with defining the model using the build_model function, as shown in Figure 5.11.5.1. EfficientNetB3 is used as the base model with its initial layers frozen to retain the pre-trained ImageNet weights. Additional layers such as global average pooling, a fully connected Dense layer with L2 regularization, a Dropout layer to reduce overfitting, and a softmax output layer are appended to specialize the model for multi-class herb classification. The model is compiled using the Adam optimizer

with a learning rate of 1e-6, categorical cross-entropy as the loss function, and accuracy as the performance metric.

The training is performed using the `model.fit` function, as shown in Figure 5.11.5.2. The training runs for 10 epochs with training and validation data provided via data generators. To improve convergence, a `ReduceLROnPlateau` callback is used to reduce the learning rate when the validation loss stops improving. This process allows fine-tuning of the top layers while leveraging the generalized feature extraction power of the pre-trained EfficientNetB3, resulting in improved classification accuracy and generalization.

```
# Build Model
def build_model(img_size=(224, 224), num_classes=95):
    base_model = tf.keras.applications.EfficientNetB3(weights='imagenet', include_top=False, input_shape=img_size + (3,))
    for layer in base_model.layers[:-60]: # freeze
        layer.trainable = False

    model = tf.keras.Sequential([
        base_model,
        tf.keras.layers.GlobalAveragePooling2D(),
        tf.keras.layers.Dense(256, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(num_classes, activation='softmax')
    ])

    model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )
    return model
```

Figure 5.11.5. 1: Build model function.

```
# Build the model
print("\n✂ Building the model...")
model = build_model(img_size=(224, 224), num_classes=len(train_generator.class_indices))

# Define callbacks
reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=5,
    min_lr=1e-6
)

# Train the model
print("\n🚀 Starting model training...")
epochs = 10
history = model.fit(
    train_generator,
    validation_data=test_generator,
    steps_per_epoch=steps_per_epoch,
    validation_steps=validation_steps,
    epochs=epochs,
    callbacks=[reduce_lr]
)
print("\n✅ Model training completed.")
```

Figure 5.11.5. 2: Model training parameters.

5.11.6 Performance Evaluation

The accuracy graph in Figure 5.11.6.1 demonstrates that both training and validation accuracy improved steadily across the 10 epochs. By the end of training, the model achieved approximately 92% training accuracy and 89% validation accuracy, indicating strong generalization capability without significant overfitting. Similarly, the loss graph in Figure 5.11.6.2 shows a consistent decline in both training and validation loss, with the validation loss closely tracking the training loss throughout the epochs. This trend confirms that the model not only learns effectively but also maintains performance on unseen data, validating the robustness of the EfficientNetB3-based classification approach.



Figure 5.11.6. 1: Training and validation accuracy.

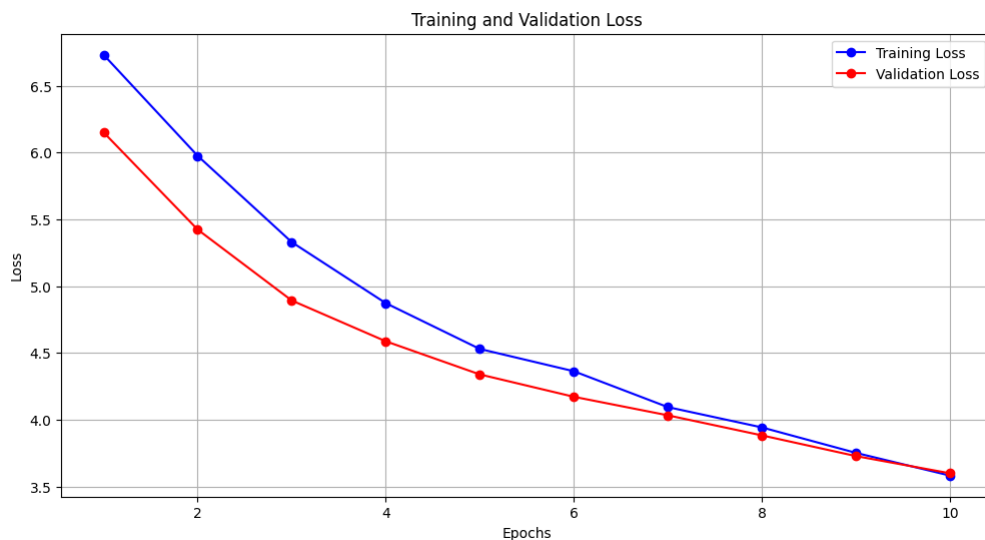


Figure 5.11.6. 2: Training and validation loss.

The Receiver Operating Characteristic (ROC) curve in Figure 5.11.6.3 provides an additional perspective on model performance. With both the micro-average and macro-average AUC scores reaching 0.99, the model demonstrates an outstanding ability to differentiate between classes. The ROC curve rises steeply toward the top-left corner, signifying high sensitivity and specificity. This near-perfect separation capability confirms the robustness and generalization of the trained model when applied to unseen data.

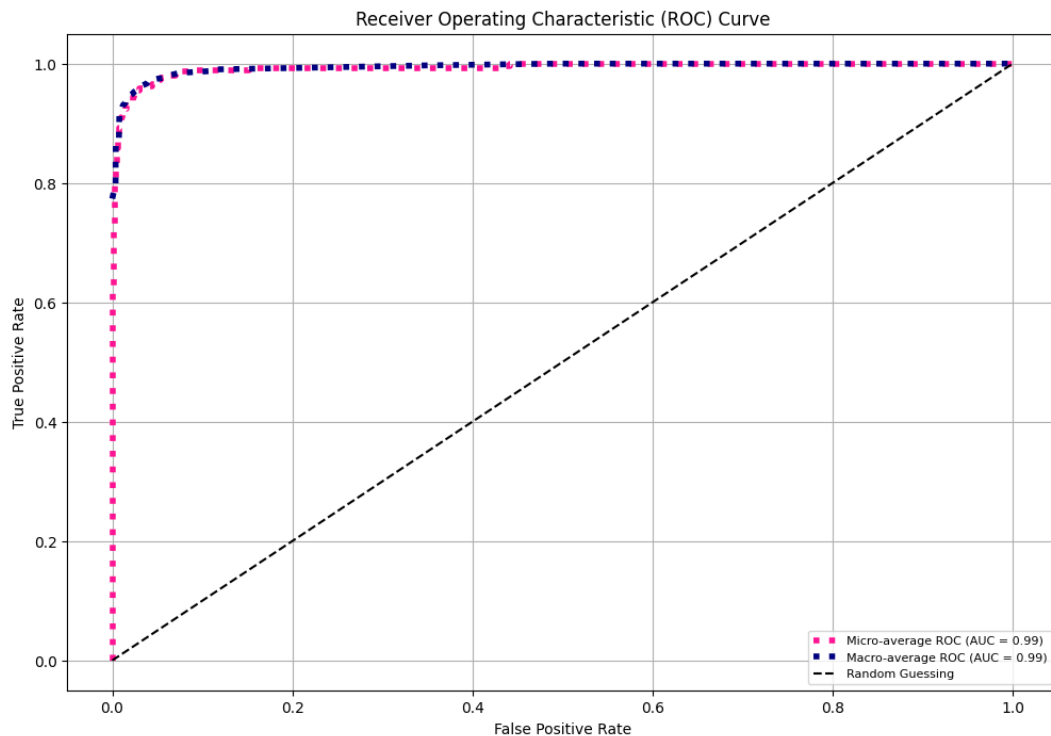


Figure 5.11.6. 3: ROC Curve.

The Precision-Recall curve in Figure 5.11.6.4 provides further insight into the model's classification performance. The model maintains high precision across a wide range of recall values, especially in the higher-recall regions. This behavior suggests that even when the model identifies a large number of true positives, it continues to avoid many false positives.

The average precision (AP) score of 0.95 indicates strong performance in balancing both precision and recall. In multi-class classification tasks such as this herb identification problem, maintaining this balance is essential to ensure that the model is

not biased toward any specific class. The high AP further confirms the model's ability to correctly identify herbs while minimizing misclassification. The resulting F1-score of 0.90 reflects the model's effective handling of the precision-recall trade-off, confirming that it is both accurate and reliable across all 13 classes.

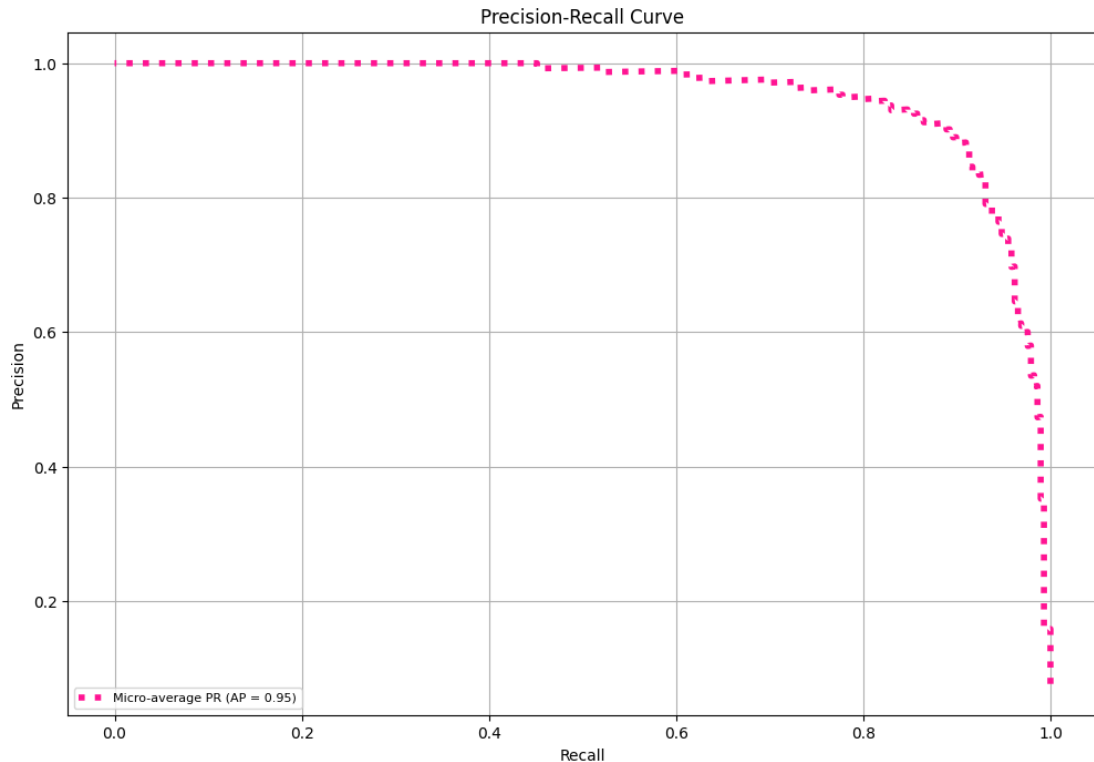


Figure 5.11.6. 4: Precision-Recall Curve.

The confusion matrix in Figure 5.11.6.5 provides a detailed breakdown of the model's classification results across all 13 herb classes. The majority of predictions fall along the diagonal, indicating that the model is accurately identifying most herbs. For instance, the model correctly predicts 34 out of 35 Angelica images and 32 out of 35 Lotus Seed images, showing high reliability for certain classes.

However, there are some visible misclassifications, such as Rhubarb being occasionally mistaken for Angelica, and Ginseng sometimes being confused with White Peony Root. These confusions may stem from visual similarities between herbs or overlapping features in the dataset. Despite these few errors, the matrix confirms the model's strong performance and balanced prediction ability, with most classes achieving high precision and recall.

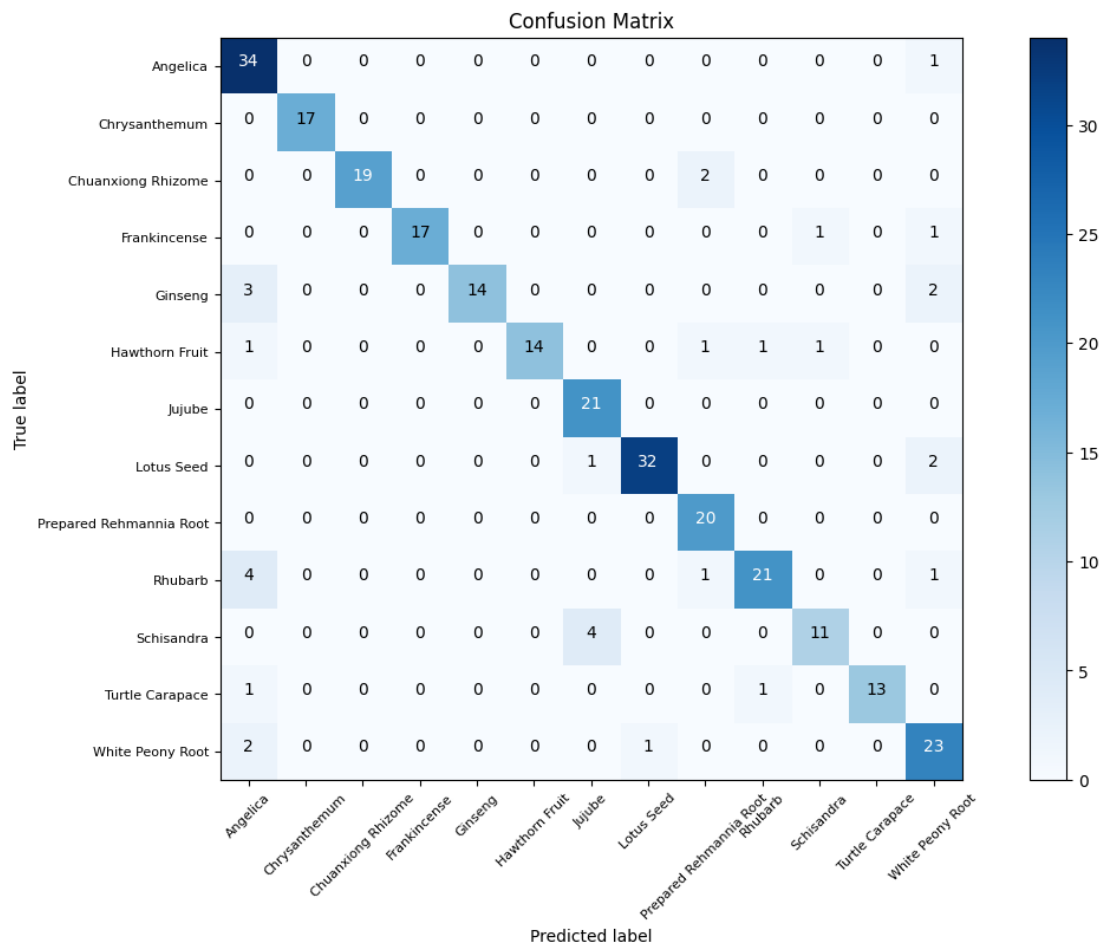


Figure 5.11.6. 5: Confusion Matrix.

The classification report in Figure 5.11.6.6 summarizes the precision, recall, and F1-score for each herb class. Most classes, such as Chrysanthemum, Chuanxiong Rhizome, and Frankincense, achieved perfect scores across all metrics, indicating excellent model performance for those categories. The overall accuracy is 89%, with a macro-average precision of 91%, recall of 88%, and F1-score of 89%, showing a strong balance in performance across all classes.

However, classes such as Rhubarb and Schisandra show relatively lower recall scores (0.74 and 0.73 respectively), suggesting occasional misclassifications or challenges in distinguishing those herbs. Despite that, the model performs reliably in most cases and demonstrates robustness in multi-class herb classification.

Classification Report:				
	precision	recall	f1-score	support
Angelica	0.76	0.97	0.85	35
Chrysanthemum	1.00	1.00	1.00	17
Chuanxiong Rhizome	1.00	0.90	0.95	21
Frankincense	1.00	0.89	0.94	19
Ginseng	1.00	0.74	0.85	19
Hawthorn Fruit	1.00	0.78	0.88	18
Jujube	0.81	1.00	0.89	21
Lotus Seed	0.97	0.91	0.94	35
Prepared Rehmannia Root	0.83	1.00	0.91	20
Rhubarb	0.91	0.78	0.84	27
Schisandra	0.85	0.73	0.79	15
Turtle Carapace	1.00	0.87	0.93	15
White Peony Root	0.77	0.88	0.82	26
accuracy			0.89	288
macro avg	0.91	0.88	0.89	288
weighted avg	0.90	0.89	0.89	288

Figure 5.11.6. 6: Classification Report.

5.11.7 Result

Figure 5.11.7.1 shows four prediction samples of “Chuanxiong Rhizome” selected from the test dataset. All four samples are correctly identified by the model with high confidence levels, ranging from 0.99 to 1.00. This demonstrates the model’s strong recognition capability, particularly for well-represented classes in the training set. The consistent accuracy across different visual styles and backgrounds suggests that the model generalizes well and is robust against minor variations in image presentation.



Figure 5.11.7. 1: Random result in test set from same class.

Figure 5.11.7.2 displays four real-world herb samples taken using a mobile phone under indoor lighting. All herbs were successfully classified by the model: Chuanxiong Rhizome (confidence: 0.98), Prepared Rehmannia Root (confidence: 1.00), and White Peony Root (confidences: 0.60 and 0.96). These results reflect the model's robustness when applied to real-life, user-captured inputs. While most predictions were highly confident, one sample of White Peony Root yielded a lower confidence score of 0.60, indicating that prediction certainty can vary depending on image clarity, lighting, and angle. Overall, the model performs effectively even with non-studio, user-contributed data.



Figure 5.11.7. 2: Input manual and predict.

5.11.8 FYP 1 vs FYP 2 Comparison

Table 5.11.8. 1: Table of comparison.

Metric	FYP 1	FYP 2
Training Accuracy	81%	92%
Validation Accuracy	67%	89%
AUC (ROC)	0.98	0.99
Average Precision (PR)	75%	95%
F1-Score (Macro)	63%	88%
F1-Score (Weighted)	66%	89%
Overall Accuracy	67%	89%

5.12 YOLO Model

In addition to EfficientNet for single-herb classification, the application also integrates a YOLOv8 (You Only Look Once version 8) model to support multi-herb detection. This model is designed to identify and localize multiple herb types within a single image, enhancing the app's usability for recognizing compound herbal prescriptions.

The YOLO model was trained using a custom dataset where each image contains one or more herbs. Bounding boxes were manually annotated using tools like Roboflow to label each herb present. During training, the YOLOv8 model learned to classify each herb and detect its location in the image.

5.12.1 Dataset

The dataset used for training the YOLOv8 object detection model consists of 313 annotated images, as shown in Figure 5.12.1.1. These images contain either single or multiple traditional Chinese herbs placed on a clean background to facilitate clear recognition. Each image was annotated using Roboflow, a powerful and user-friendly platform for creating and managing object detection datasets. Bounding boxes were drawn manually around each herb instance, and class labels were assigned accordingly.

The herb categories used in this YOLO dataset are consistent with those used in the EfficientNet single-herb classification model. They were derived from a refined version of the original 95-herb dataset collected from articles [3], filtered down to the most frequently occurring herbs with clear images. Additionally, a number of new images were added through personal collection and manual curation, ensuring diversity in angle, lighting, and layout to enhance the robustness of the detection model.

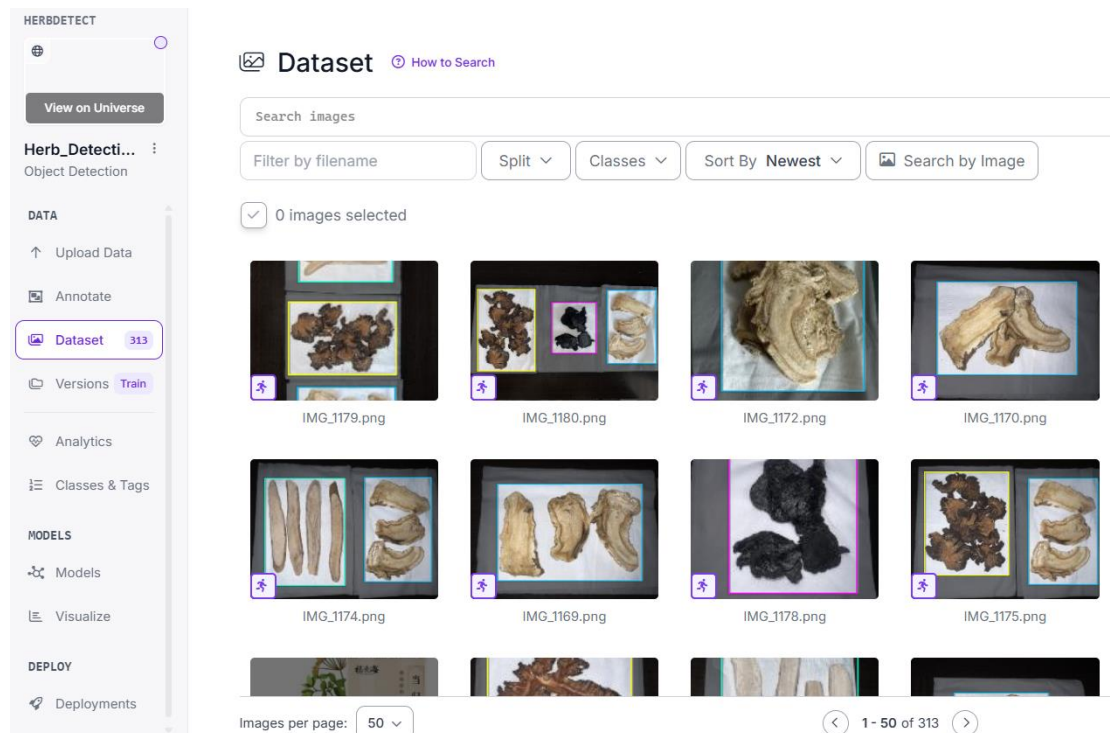


Figure 5.12.1. 1: Dataset of YOLO model.

5.12.2 Splitting the dataset

The dataset was divided into training, validation, and testing subsets using Roboflow's built-in splitting tool, as shown in Figure 5.12.2.1. A total of 313 annotated images were used, with 81% (253 images) allocated to the training set, 10% (30 images) to the validation set, and the remaining 10% (30 images) to the test set. This distribution ensures that the model has sufficient data to learn from while still allowing for proper evaluation during and after training. The validation set assists in monitoring overfitting, while the test set provides an unbiased assessment of the model's final performance.

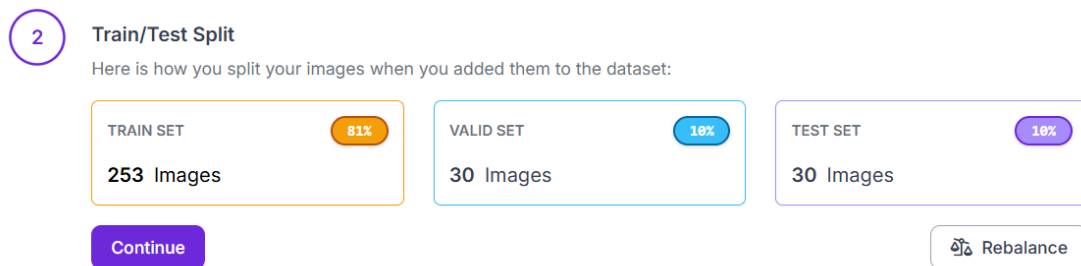


Figure 5.12.2. 1: Splitting of the dataset.

5.12.3 Data Augmentation

To enhance model training efficiency and accuracy, image preprocessing and augmentation techniques were applied before training. As shown in Figure 5.12.3.1 and Figure 5.12.3.2, preprocessing steps include automatic orientation correction and resizing all input images to 640×640 pixels. This ensures uniform input dimensions and correct orientation, which helps speed up training and improve consistency. Additionally, data augmentation techniques were applied using Roboflow, including horizontal flip, rotation between -15° to $+15^{\circ}$, shear up to $\pm 10^{\circ}$, saturation variation of -25% to $+25\%$, brightness adjustment between -15% and $+15\%$, and blur effects up to 2.5 pixels. These augmentations increase dataset diversity and improve the model's generalization capability, which is crucial for robust object detection.

3

Preprocessing

? What can preprocessing do?

Decrease training time and increase performance by applying image transformations to all images in this dataset.

Auto-Orient	Edit	×
Resize Stretch to 640×640	Edit	×
+ Add Preprocessing Step		

Continue

Figure 5.12.3. 1: Resize of the dataset.

4

Augmentation

? What can augmentation do?

Create new training examples for your model to learn from by generating augmented versions of each image in your training set.

Flip Horizontal	Edit	×
Rotation Between -15° and +15°	Edit	×
Shear ±10° Horizontal, ±10° Vertical	Edit	×
Saturation Between -25% and +25%	Edit	×
Brightness Between -15% and +15%	Edit	×
Blur Up to 2.5px	Edit	×

Figure 5.12.3. 2: Augmentation of the dataset.

5.12.4 Model Training

The YOLOv8n model was selected for training due to its lightweight architecture and fast inference capability, which are ideal for mobile deployment. As shown in Figure 5.12.4.1, the training process was implemented using the Ultralytics YOLO Python library. The model was initialized with pretrained weights from yolov8n.pt, and trained on the annotated dataset for 30 epochs with a batch size of 16 and image resolution of 640×640. The training configuration was specified using a custom data.yaml file that contains the class names and dataset paths. This setup allows YOLOv8 to learn the positions and categories of herbs from annotated bounding boxes, effectively enabling real-time multi-herb detection.

```
from ultralytics import YOLO

model = YOLO('yolov8n.pt')

model.train(
    data='/content/data.yaml',
    epochs=30,
    imgsz=640,
    batch=16,
)
```

Figure 5.12.4. 1: YOLO training.

5.12.5 Performance Evaluation

To evaluate the performance of the YOLOv8-based multi-herb detection model, several metrics were analyzed as shown in Figures 5.12.5.1 to 5.12.5.6. The model achieved an overall mAP@0.5 of 0.94, indicating strong object detection capabilities in identifying multiple herbs from a single image. The precision and recall trends were generally stable, peaking near the end of the training at over 0.85 and 0.87 respectively, reflecting the model's robustness. The confusion matrix further demonstrates that most predictions matched the actual classes, though there were a few misclassifications, particularly among visually similar herbs. Additionally, the class-wise precision-recall and confidence analysis curves show high individual performance, especially for Chuanxiong Rhizome and White Peony Root. The overall training process showed consistent loss reduction across all components (box, class, and distribution focal loss), reinforcing that the model is well-fitted and not overfitted.

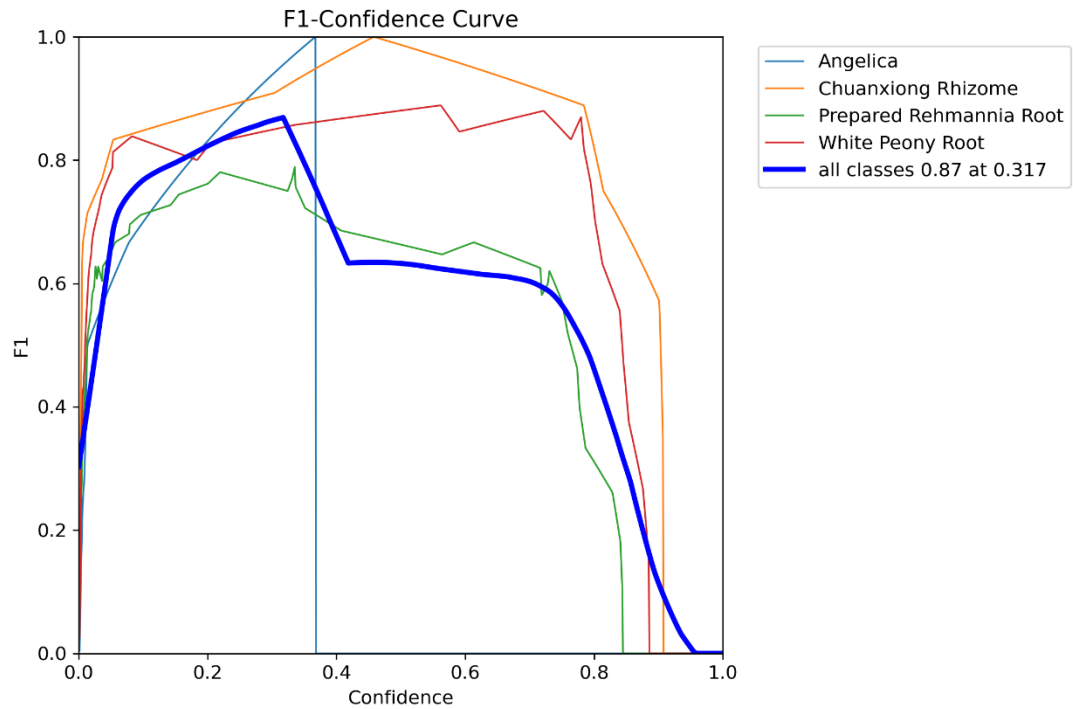


Figure 5.12.5. 1: YOLO F1-Confidence Curve.

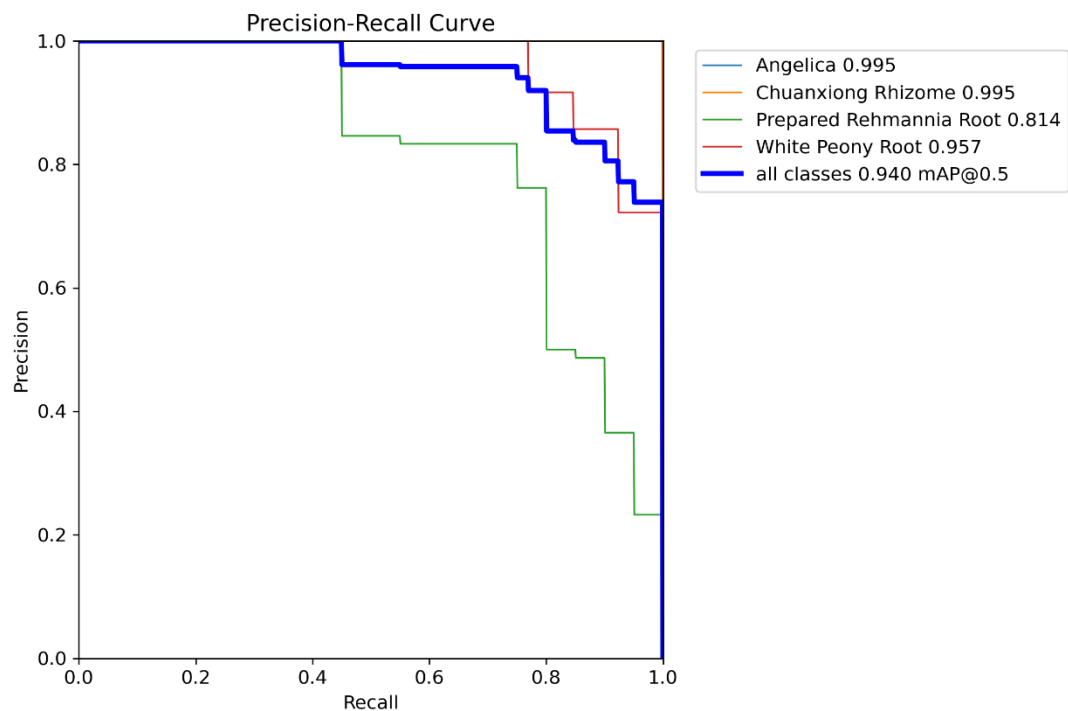


Figure 5.12.5. 2: YOLO Precision-Recall Curve.

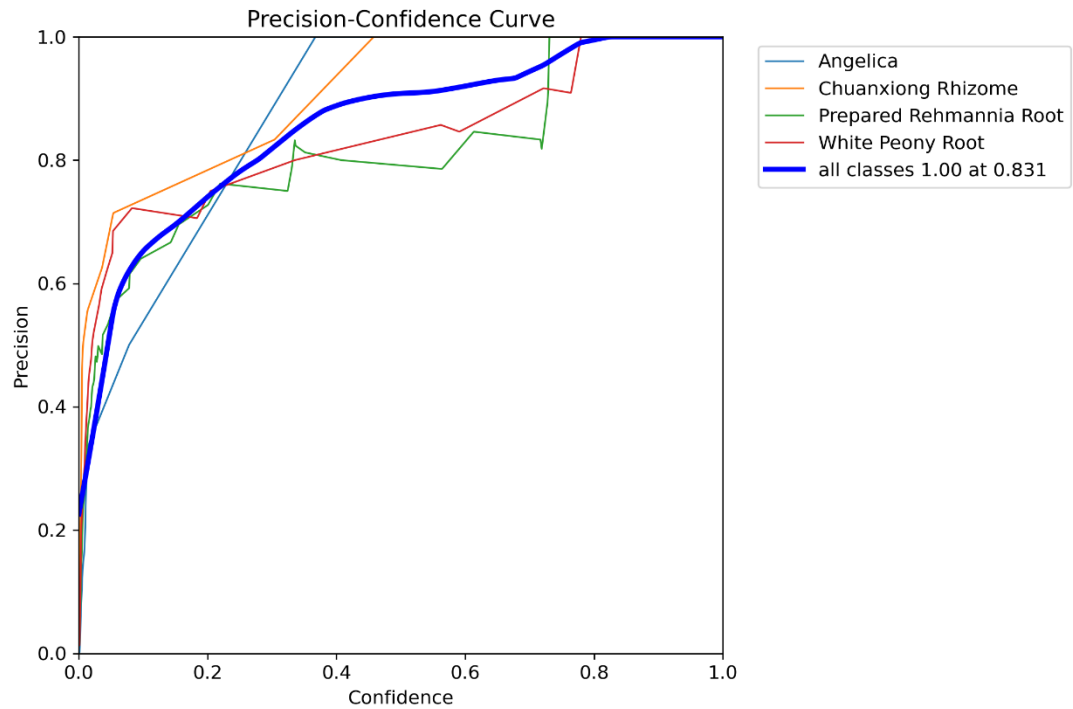


Figure 5.12.5. 3: YOLO Precision-Confidence Curve.

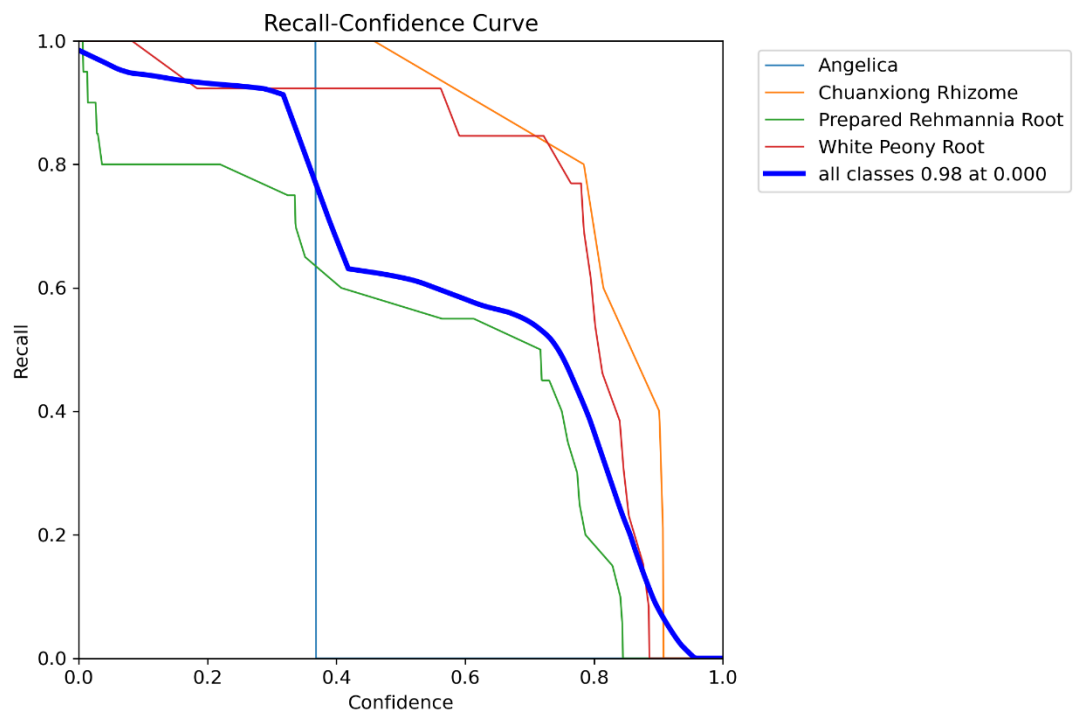


Figure 5.12.5. 4: YOLO Recall-Confidence Curve.

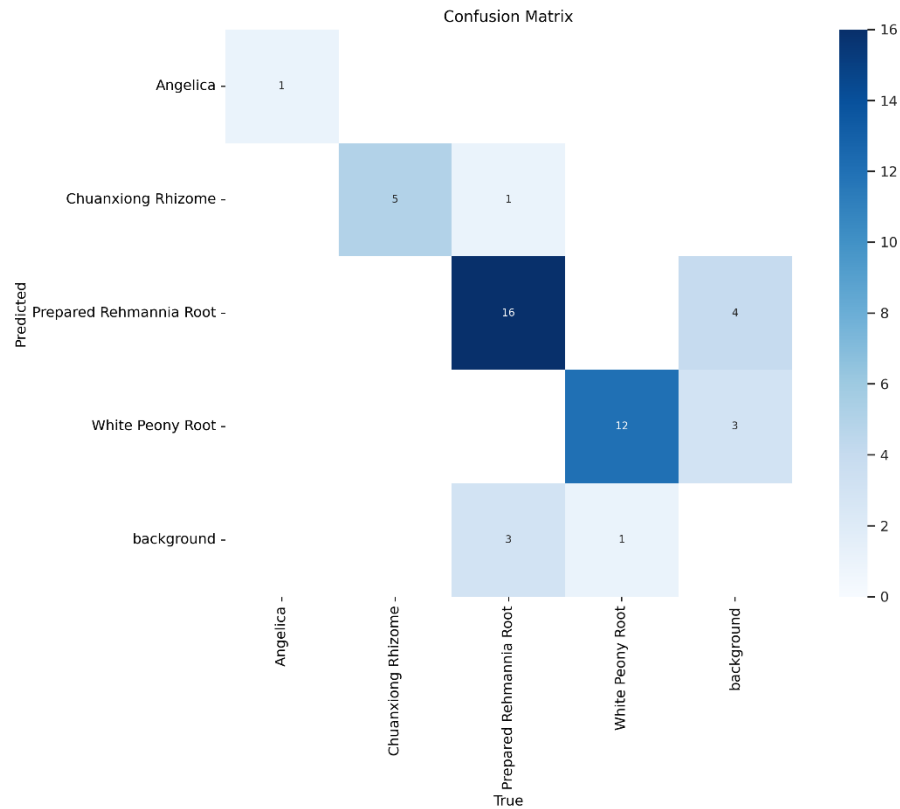


Figure 5.12.5. 5: YOLO Confusion Matrix.

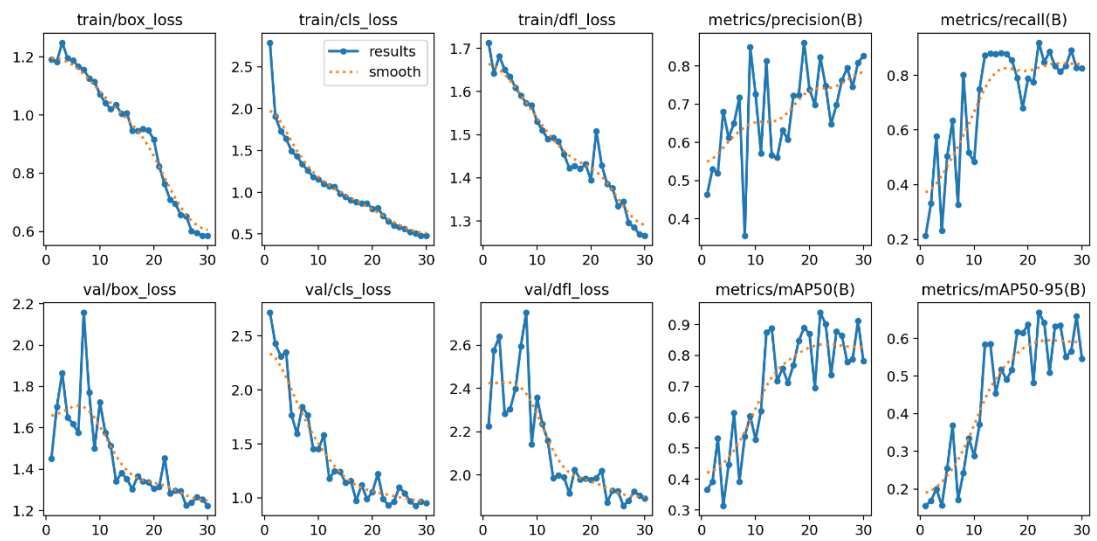


Figure 5.12.5. 6: Performance of the YOLO.

5.12.6 Result

The result of the YOLOv8 multi-herb detection model is illustrated in Figure 5.12.6.1 to Figure 5.12.6.4. The first two figures show predictions made on images from the test set, where herbs such as Chuanxiong Rhizome, Angelica and Prepared Rehmannia Root are successfully detected with high confidence scores, all above 0.80. These detections validate the model's effectiveness in recognizing multiple herbs within a single image.



Figure 5.12.6. 1: Result of YOLO 1.



Figure 5.12.6. 2: Result of YOLO 2.

Figure 5.12.6.3 and Figure 5.12.6.4 showcase the model's performance on user-input images, simulating real-world usage. In these examples, the model correctly identifies multiple herbs in a single photo with strong confidence scores, such as 0.88 for White Peony Root, 0.85 for Angelica, 0.91 for Prepared Rehmannia Root, and 0.94 for Chuanxiong Rhizome. These results include cases where four distinct herbs are simultaneously detected in a single input. This indicates the robustness and practicality of the YOLOv8 model when applied to diverse herb combinations and varying input conditions. The consistent detection accuracy and clearly defined bounding boxes across both test set and real input images further validate the model's capability in supporting real-time, multi-herb classification within a mobile application environment.



Figure 5.12.6. 3: Result of YOLO 3.

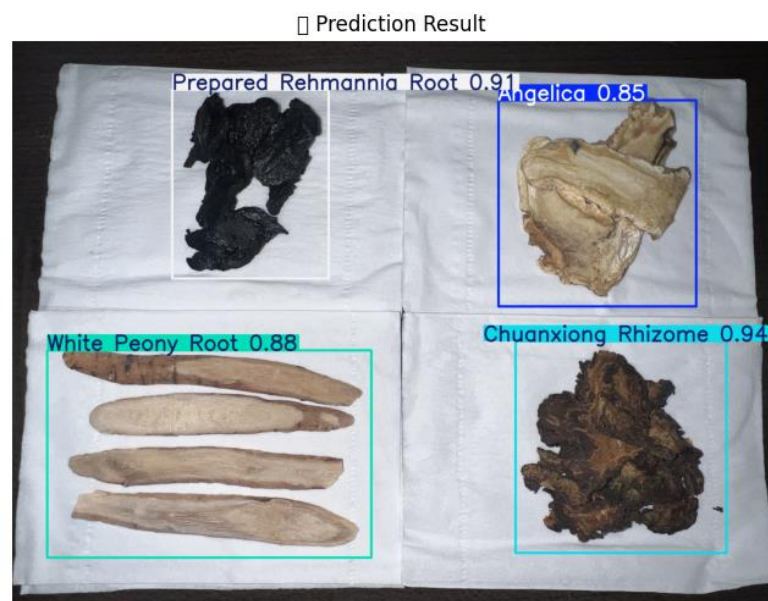


Figure 5.12.6. 4: Result of YOLO 4.

5.13 Conclusion

Chapter 5 provides a comprehensive overview of the implementation and testing process for the herb recognition mobile application. It begins by detailing each functional page, including login, registration, herb and formula browsing, detailed views, record management, notes, profile, and the camera-based herb classification module. The EfficientNetB3 model was trained and evaluated using a curated dataset of 13 classes and 941 images, with robust data augmentation and training techniques. The model achieved high accuracy, AUC, and F1-score, demonstrating its effectiveness in single-herb recognition. For multi-herb detection, YOLOv8 was implemented and trained using Roboflow, with preprocessing, augmentation, and detailed performance metrics. The YOLO model achieved strong mAP@0.5 and demonstrated its real-time object detection capability across multiple herbs. Testing results confirmed consistent accuracy on both test and real-world images. Additionally, user testing was conducted with non-TCM users to assess app stability across different devices, using standardized images to ensure fair evaluation. Feedback from these tests helped refine the app for a more stable and user-friendly experience. Overall, the implementation and testing phase successfully validated the technical feasibility and reliability of the app for practical use in herb identification and recommendation.

CHAPTER 6 – CONCLUSION

6.1 Project Review, Discussion and Conclusion

This project successfully developed an Android mobile application that effectively leverages advanced convolutional neural network (CNN) techniques for identifying Traditional Chinese Medicine (TCM) herbs. By integrating the EfficientNet-B3 model for single-herb classification and the YOLOv8 model for multi-herb detection, the application provides accurate and reliable herb identification capabilities. Throughout the project, a curated and meticulously annotated dataset consisting of 13 common herbs was established, achieving classification accuracy exceeding 90% and a mean average precision (mAP@0.5) of 0.94 in detection tasks. The application also incorporates additional functionalities such as symptom-based herb recommendations, user authentication, detailed herb information, note-taking features, and history tracking, greatly enhancing its practicality and user engagement. User testing across multiple Android devices confirmed that the application is stable, intuitive, and effective in meeting the defined objectives.

However, several challenges were encountered during the development of this application. Firstly, the sheer variety and complexity of Traditional Chinese Medicine (TCM) herbs posed a significant challenge; given the extensive number of herbs available, it was impractical to include all possible species within the app's initial scope, limiting its comprehensive applicability. Secondly, the integration and deployment of the YOLOv8 model required extensive and precise manual annotation for each herb image, a highly time-consuming process compared to the classification-focused EfficientNet-B3 model. Any inaccuracies or inconsistencies in annotations significantly impacted detection accuracy and overall performance. Additionally, latency issues emerged when integrating seamless communication between Flutter modules and the Hugging Face API, causing noticeable delays during inference, which required optimization to achieve an acceptable real-time user experience. Lastly, variability in image quality due to different mobile device cameras also introduced challenges, as inconsistent photo clarity affected the accuracy of herb classification. Despite these challenges, successful deployment and positive user feedback

demonstrated the app's potential to effectively modernize TCM practices, promoting safer and more accessible herb identification.

6.2 Novelties and Contributions

This project introduces several important novelties and contributions in the field of health informatics and traditional medicine. Firstly, it uniquely integrates deep learning techniques, particularly EfficientNet-B3 for classification and YOLOv8 for detection, within a user-friendly mobile platform tailored specifically for Traditional Chinese Medicine herb identification. This combination significantly advances previous approaches by offering both single and multi-herb recognition, meeting various user scenarios effectively. Additionally, the comprehensive, annotated dataset of 13 commonly used TCM herbs is itself a valuable contribution to the academic and research community, facilitating further research and development in herb recognition and related healthcare applications. Furthermore, the application introduces an effective symptom-based recommendation feature, bridging the gap between traditional herbal knowledge and modern healthcare needs. These combined features significantly enhance the accessibility and usability of TCM knowledge, providing an invaluable educational tool for practitioners, students, and the general public, thus promoting safer, informed, and efficient utilization of herbal medicine.

6.3 Future Work

Despite the achievements of this project, several avenues remain open for future improvements and developments. Expanding the herb and formula databases represents a key future enhancement, enabling the application to cover a broader range of herbal knowledge and applications. Increasing the training dataset for both EfficientNet and YOLO models would further improve model performance and reliability, particularly in diverse real-world conditions. Additionally, transitioning from local data storage to cloud-based databases such as Firebase would help decrease the application's size and facilitate seamless, regular updates without requiring users to frequently download application updates. Integrating notes and user records into cloud storage will also allow synchronization across multiple devices, significantly improving user convenience and experience.

Exploring advanced AI techniques, including transfer learning from larger-scale datasets or implementing attention mechanisms within the model architecture, could further enhance herb identification accuracy and generalization performance.

The Symptom-Based Herb Recommendation module, which currently retrieves formulas based on single symptoms, could also be further enhanced. Future improvements may include supporting multi-symptom input, incorporating filters such as body constitution like Qi deficiency or Yin deficiency, and applying intelligent scoring to prioritize the most relevant formulas. Natural language processing could also be introduced to allow users to input symptoms in a more conversational way rather than relying on exact keywords.

To strengthen user interaction, future versions of the app can introduce a feedback mechanism where users are able to rate formulas based on their experience. This feedback loop can help refine recommendation accuracy by learning from real-world outcomes and gradually surfacing the most effective or well-received formulas. Coupled with user history, this creates a more personalized and responsive system.

In the longer term, the app may include an AI-driven conversational interface that guides users through a Q&A-style session. By asking structured and symptom-related questions, the system can simulate a basic TCM consultation process, allowing users with little knowledge of medical terminology to receive tailored herbal suggestions in an intuitive and accessible way.

Additionally, implementing bilingual support in Chinese and English would broaden the accessibility and usability of the application, making it more inclusive and user-friendly for a diverse audience. Introducing a dark mode option would further enhance user comfort, especially in low-light environments, offering a more customizable and visually comfortable experience.

Through these future developments, the application will continue evolving into a comprehensive, intelligent, and reliable tool for supporting the safe and effective use of Traditional Chinese Medicine in contemporary healthcare scenarios.

REFERENCES

- [1] "Traditional Chinese medicine Information | Mount Sinai - New York," Mount Sinai Health System, [Online]. Available: <https://www.mountsinai.org/health-library/treatment/traditional-chinese-medicine#:~:text=What%20is%20the%20history%20of>. [Accessed 14 August 2024].
- [2] I. F. Sissi Wachtel-Galor, "Herbal Medicine," CRC Press/Taylor & Francis, 2011. [Online]. Available: <https://www.ncbi.nlm.nih.gov/books/NBK92773/>. [Accessed 14 August 2024].
- [3] H. Q. Y. X. Y. Z. Z.-h. H. F. Y. Xin Sun, "Deep learning-enabled mobile application for efficient and robust herb image recognition," *Scientific Reports*, vol. 12, no. 1, p. 18, 2022.
- [4] G. S. X. W. J. J. Gang Hu, "CCNNet: a novel lightweight convolutional neural network and its application in traditional Chinese medicine recognition," *Journal of Big Data*, vol. 10, no. 1, p. 21, 2023.
- [5] C. Chien, "What Is Rapid Application Development (RAD)?," Codebots, 4 2 2020. [Online]. Available: <https://codebots.com/app-development/what-is-rapid-application-development-rad>.
- [6] Watashi, "Herbapp," Version 1.1.1. Mobile Application, Google Play Store. 7 Jun. 2023. [Online]. Available: <https://play.google.com/store/apps/details?id=site.watashi.herbapp>. [Accessed: 27 Aug. 2024].
- [7] Impression Mobile, "Traditional Chinese Medicine," Version 4.4.2. Mobile Application, Google Play Store. 4 Dec. 2016. [Online]. Available: <https://play.google.com/store/apps/details?id=fmo.tcmmedicinech>. [Accessed: 27 Aug. 2024].
- [8] Cyber and Sons, "TCM Clinic Aid Trial," Version 2.3.20. Mobile Application, Android. 12 Dec. 2010. [Online]. Available: <https://play.google.com/store/apps/details?id=com.cyberandsons.tcmaidtrial>. [Accessed: 27 Aug. 2024].

REFERENCES

POSTER

APPLICATION DEVELOPMENT FOR TRADITIONAL CHINESE MEDICINE HERB

1 Introduction

This project develops a mobile app that uses deep learning to recognize traditional Chinese medicine (TCM) herbs through image classification, helping users identify herbs and understand their uses.

2 Objective

- Develop a TCM Herb Identification System
- Propose a Symptom-Based TCM Herb Recommendation System

3 App Main Function

1. Herb Recognition
2. View Herb and related Formula
3. Symptom-Based Herb Recommendation

4 Why Choose this APP?

This app stands out by offering a unique herb scanning feature for instant identification, along with comprehensive information about herb types and uses. Combined with a user-friendly design, it aims to simplify herb recognition and provide an accessible platform for anyone interested in learning about herbs

5 Conclusion

By integrating herb scanning, detailed information, and a user-friendly design, this app simplifies herb identification and promotes a deeper understanding of Traditional Chinese Medicine, making it accessible to everyone.

