Plant Disease Detection Mobile System

COMP826 Mobile Systems Development 2023 S2
Milestone 2: Prototype, Evaluation and Recommendations

Prof. Roopak Sinha
Dr. Matthew Kuo

WULAN E

22169691

Computer and Information Science Auckland University of Technology
Auckland, New Zealand

Oct 29, 2023

Table of Contents

1 Overview

1.1 Background of the topic

1.1.1 Description of the topic

The agriculture industry plays an important role in generating revenue and meeting the food demand of the people (Pantazi et al., 2018). Often a country is unable to supply enough food to meet demand due to disease in the agriculture sector. The traditional method of identifying crop diseases is challenging for reliable crop evaluation. The traditional approach to identifying crop diseases begins with the employment of a domain specialist who visits the site and observes the crop using optical inspection. This is a time-consuming and labor-intensive technique. It also necessitates constant crop monitoring. Farmers in some areas do not have access to experts, which is another major issue. Optical tracking is not always reliable for detecting crop diseases. Using traditional methods to diagnose the disease in the crop will not provide an accurate assessment (Memon et al., 2022). In such situations, deep learning is typically used because it allows the computer to autonomously learn the most suitable feature without human intervention. An initial attempt to use deep learning for image-based plant disease diagnosis was reported in 2016, where the trained model was able to classify 14 crops and 26 diseases with an accuracy of 99.35% against optical images (Mohanty et al., 2016).

1.1.2 Description of project details and its justification

I proposed a plant disease detection system, which can not only detect plants and give information related to plants, but also detect plant diseases and give corresponding maintenance methods and suggestions. This system can be installed on any mobile device, allowing users to use the camera to capture images of plants or enter plant names directly. The images or input information will be uploaded to the server, where the information will be processed. Once processing is complete, the server will send the results back to the user's device. The results could be one or multiple plant matches. Users can then determine which one meets their needs based on the provided results. The system also enables users to save the query results to "My Plant," facilitating future reference and ongoing tracking of plant statuses.

1.2 Project description

1.2.1 Features, role, and scope

Novel Features of the Plant Disease Detection is the disease detection and My plant function. It employs image recognition technology to detect plant diseases, offering an innovative approach. Users can obtain accurate diagnosis results by simply capturing or uploading photos, without requiring specialized botanical knowledge. And achieves rapid identification and provides detailed diagnostic reports in a short period, allowing for timely action to prevent disease spread.

In Milestone 1, the key functional requirements for the VR Grocery Store application are as follows:

- F1 - User Functional Requirement: User authentication and authorization. Such as:
  User login/logout,
  Forgot password,
  Create/Edit User account.
  Edit My plant
- F2 - Search Functional Requirement: Users can swiftly obtain search results for plants or plant diseases. Such as:
  Search plant or plant disease.

- F3 - Disease detection functional Requirement: Users can swiftly obtain detection results for plant diseases with high accuracy.
  Detect plant disease.

Due to the limited development time available and my software developer background, I will take on the role of being a developer to produce and complete the system. I focus on the above three Features (F1, F2, F3). However, due to time constraints, not all the above features were implemented. The following features have been implemented so far:

- F1 - User Functional Requirement: User authentication and authorization. Such as:
  User login/logout,
  Create/Edit User account.
- F3 - Disease detection functional Requirement: Users can swiftly obtain detection results for plant diseases with high accuracy.
  Detect plant disease.

In Milestone 2, I'll cover the implemented functionality in detail, including the development process, design patterns, specific code implementations and algorithms, and the development tools used. Finally, I will summarize the experience of the entire development process and make suggestions for future improvements.

Github link: https://github.com/ewulan/daoctor.git
Youtube link: https://youtube.com/shorts/uCxy95fmTB0?feature=share

1.2.2 Existing studies on the topic

Several machine learning techniques are already being used to detect plant diseases. Such as Vector Machine (SVM) Classification Technique, Artificial Neural Network (ANN) Classification Technique, and Convolutional Neural Network Classification methods (Shruthi, Nagaveni, & Raghavendra, 2019).

In 2016, Padol et al. conducted a study on the detection of diseases in citrus trees, specifically those affected by canker and anthracnose diseases such as grapefruit, lemons, lime and oranges. The results showed a high rate of accurate identification at 95% (Padol & Yadav, 2016).

Pawar et al. conducted a study utilizing the neural network Back propagation method to identify cercosporin (leaf spot) disease in groundnut plants. Through their experiments and observations, they successfully classified four types of diseases from 100 sample diseased leaf images, achieving an accuracy rate of 97.41% (Pawar & Jadhav, 2017).

Ferentinos devised a CNN classification method to identify crop diseases, utilizing a dataset comprising 87848 images of 25 distinct plant species affected by 58 different diseases. The approach yielded an impressive accuracy rate of 99.53% (Ferentinos, 2018).

2 System artefacts

2.1 Development artefacts of the system

2.1.1 Development process

In Milestone 1, the development process I have chosen for this project is a hybrid agile model, i.e. a combination of Scrum and Waterfall models. All the requirements have been analyzed in Milestone 1 and will not be changed at a later stage. The waterfall model is used for the requirements part as it

ensures the stability of the requirements before the start of the software project, which is an advantage of the waterfall model (Yahya and Maidin, 2022).

Therefore, the requirements as well as the waterfall model will not be discussed here. Let's move directly to the code development process and discuss the Scrum model that needs to be adopted for the code development process. As already mentioned in Milestone 1, Scrum has attracted the interest of practitioners because the technique facilitates active communication with product owners and stakeholders so that projects can be developed, adapted, and achieve their goals (Yahya and Maidin, 2022).

Considering that no other users are involved in user testing during the development process, to save development time, the installation files are not continuously deployed to all devices. Nor do I deploy and test the product in the cloud. To improve development efficiency, all code development and testing are done within a single local development environment.

Considering that the whole development process is done by me alone and the development time is limited. Therefore, some parts of the Scrum development have been removed, especially the parts involving multi-party cooperation and communication. For example, the Meeting section, including Planning Meeting, Retrospective Meeting, and Daily Scrum Meeting, was removed from the Scrum development. The alternative approach is to start each workday by reviewing the existing development tasks and verifying with oneself the development and testing tasks to be completed today. This ensures the avoidance of redundant work and keeps the requirements up to date. This process helps improve work efficiency and allows for the continuous updating of project requirements to meet customer demands. The reworked Software Development Life Cycle (SDLC) is shown in Figure1:

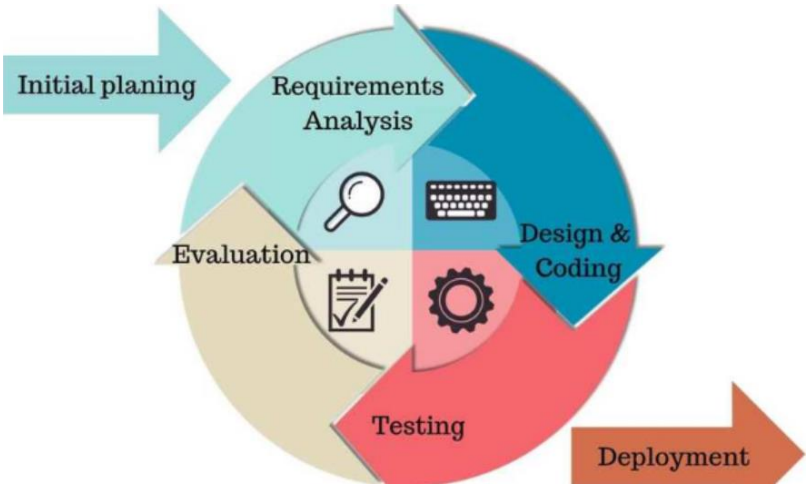

Figure 1: Modified version of the SDLC

As shown in Figure 1 above, the main tasks are as follows:

- Requirement analysis: Clarify and subdivide the requirements, which must be clear. Preparation for code development.
- Design and Coding: Starting to write code to implement the requirements. The goal of coding is to turn requirements into actual executable software.
- Testing and modification: The code written needs to be validated and tested to ensure that it meets the requirements definition. If it does not meet the requirements, modify the code.

- Evaluation: Reviewing the performance of the previous Sprint, identifying problems and opportunities for improvement, and developing a plan to improve the next Sprint. this is an opportunity for self-assessment and improvement.

In Milestone 1, each feature serves as a sprint, aligning with F1, F2, and F3 to generate three distinct sprints. Each sprint entails Requirement analysis, Design and Coding, Testing and modification, and Evaluation tasks. Upon completion of each sprint, an independent functional requirement is generated. At the end of each sprint, a Product Increment is submitted that is functionally complete and quality assured and does not cause regression issues with the previous product.

The Scrum methodology has its own iterative, fast-iterating characteristics, as well as the requirement for greater interaction between individuals, leading to a natural fit with certain software development methods. For example, Test Driven Development (TDD) starts with writing test cases, which defines the inputs and outputs of the functionality to be developed. Then you write the functional code and run the corresponding test cases after completing the development. If the partial test fails, the code is modified. If the partial test passes, run all the test cases to determine if there are any regression problems, and if there are, modify them in time. Refer to Figure2 for the flowchart.

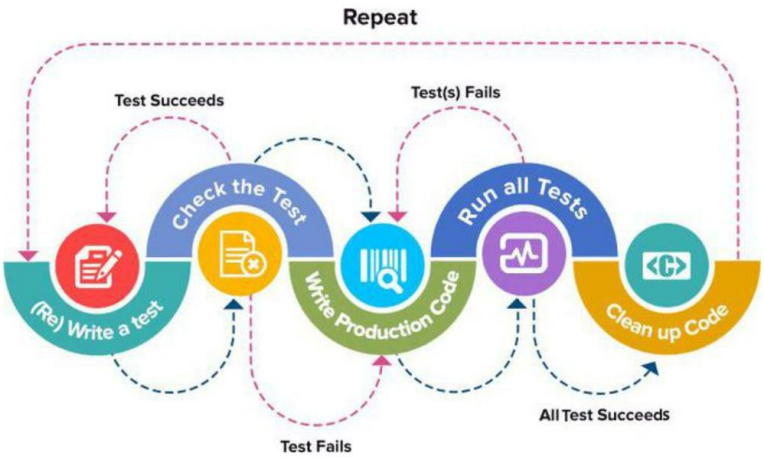

Figure 2: TDD

In milestone 1, the test case has been designed based on the requirements (F1, F2, F3). Integrate the TDD into the scrum as follows:

- Requirement analysis: Check or rewrite the tests.
- Design and Coding: Write production code.
- Testing: Run all the tests.
- Evaluation: Submit and clean up code.

Combined scrum and TDD together and applied on a software development process. The designed code is clearer, which makes it easier to maintain and flexible. And time is utilized more efficiently. Because testing is done at the same time, changes to fix bugs are also done at the same time (Aggarwal and Singhal 2019).

An example of using scrum in conjunction with a TDD approach is when, during the development of F3, one of the user actions is found to be completely unnecessary. In the original Milestone 1 design

(Figure 5: Sequence Diagram for disease detection), the user uploads a picture and waits for the plant disease to be detected. In the original Milestone 1 design (Figure 5: Sequence Diagram for disease detection), during the process of uploading a picture and waiting for the plant disease detection, the user needs to do the confirm operation to make sure that the plant returned by the system is the plant to be detected, and if the user clicks “confirm”, then the plant disease detection will be carried out, or else the user need to upload the picture again. During the testing process, we found that this middle step is not necessary at all, and the goal of the test is Test case 9: when users utilize authentic plant images, the system provides detection results. I found that the problem does not conform to Test case 9, so I immediately made a Code modification. The modified process is after the system confirms that the plant image is plant, the system directly detects the disease without the need for intermediate user confirmation to get the disease detection results. scrum combined with the TDD approach can help developers find and solve problems earlier and save development time.

2.1.2 Technology

As the core function of this project is plant disease detection, and this function in the need to use deep learning technology. Therefore, Python is used as the main development code on the server side. J Brownlee think Python is the best-of-breed platform for getting started and very quickly developing powerful and even state-of-the-art deep learning models in the Keras deep learning library for Python. Unlike R, Python is a fully featured programming language allowing you to use the same libraries and code for model development as you can use in production. Unlike Java, Python has the SciPy stack for scientific computing and scikit-learn which is a professional grade machine library (Brownlee, 2016).

The front-end mainly follows web development standards. The main reason for the inconsistency with the initial technology Flutter of milestone 1 is that the development time is limited, and if we follow the original technology, I cannot complete the development and testing of three modules at the same time in a short period of time, including the server, the client, and the communication between the server and the client. Considering that the front-end is only for demonstration purposes, the front-end was replaced with web development to reduce the time spent on code development.

The programming language used for web development is JavaScript, accompanied by markup text language html and style rendering CSS. web application development is based on the browser, the browser itself has solved the problem of multi-platform compatibility, so web development is generally no need to consider cross-platform compatibility issues faced.

To reduce the technology stack and save time, flask, a lightweight web framework, is used. flask uses the Python language. flask provides routing functionality that allows users to map different URLs to specific handler functions that handle different types of HTTP requests (GET, POST, etc.). So users can write these handlers to respond to client requests. Flask also has database support, which you can use to interact with databases, retrieve data from them, store it, or perform other database operations. Flask integrates with different types of databases, including relational databases (e.g., MySQL, PostgreSQL) and NoSQL databases (e.g., MongoDB). Copperwaite and Leifer (2015) describe SQLAlchemy as an extremely powerful library for working with relational databases in Python in their book "Learning Flask Framework". Instead of writing SQL queries by hand, we can use normal Python objects to represent database tables and execute queries (Copperwaite, M., & Leifer, C. 2015).

2.1.3 Logical decomposition

In the actual development process, the MVVM architecture continues to be used at the logical level, consistent with milestone 1. MVVM is the more suitable design model for this mobile system application. MVVM provides better code maintainability by separating the different concerns of the application, as well as better testability due to the use of data binding mechanisms in the design pattern.

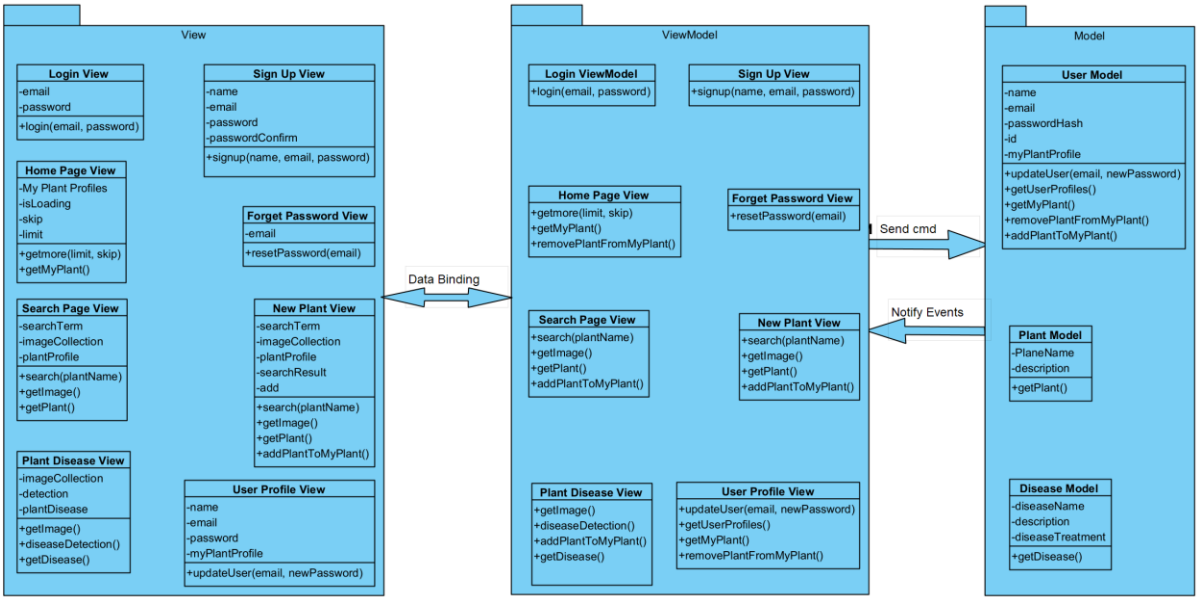

Figure 3: MVVM Class Diagram in milestone 1

Based on the design of milestone 1(Figure 3), due to time constraints, the actual development is as follows:

- Model: In the milestone 1 design, there are three data models, namely User data model, Plant data model and Disease data model. In the final version, there are only two, namely User data model and Plant data model. The previously designed Disease data model was merged into Plant data model. The main reason is that during the development process, it was found that many plant diseases are closely related to specific plant species, and a certain plant disease may not be common to all plants. Therefore, the two data models have been merged.

- View: The following Views are currently implemented:
  i. Login View
  ii. Sign Up View
  iii. Home page View
  iv. User Profile View
  v. Plant Disease View

- ViewModel: The following ViewModules are currently implemented:
  i. Login ViewModle: User authentication
  ii. Sign Up ViewModle: Add new user account.
  iii. Home page ViewModle: Navigation links or menus
  iv. User Profile ViewModle: Allows users to modify their own profile

v. Plant Disease ViewModle: Uses deep learning technology to detect diseases in the images uploaded by users. Currently, only 8 diseases of rice are supported.

Each of these functional components can communicate with each other. The implementation of communication relies heavily on the RESTful API in the Flask architecture, which is a web service that uses HTTP requests to interact with data, and is used for communication between mobile applications, front-end and back-end systems. RESTful APIs can also be used for database interaction, allowing operations to read or modify data.

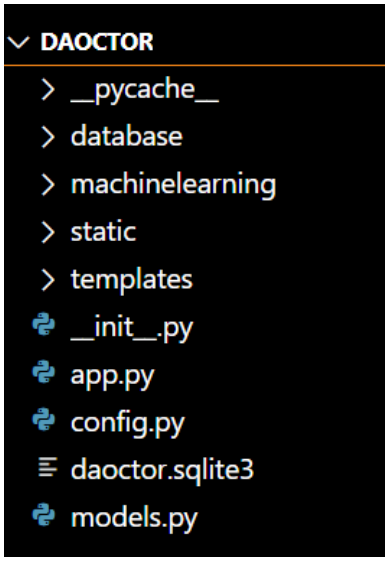

Figure 4: MVVM Code Structure

As shown in Figure 4, the templates and static folders correspond to all View-related implementation code, the database folder and models.py corresponds to all Model-related databases, and the Machinelearning folder and other files correspond to ViewModel-related implementation code.

Flask creates a RESTful API that enables the View and ViewModel to send requests to the server to fetch, create, update, or delete data using standard HTTP request methods such as GET, POST, PUT, and DELETE. Flask implements the ViewModel and Model communication through the extension library Flask-SQLAlchemy. Model communication, mainly handling requests, validating data, connecting to databases, and implementing authentication and authorization. These are the features needed to create a robust RESTful API.

Development based on the Flask architecture fully embodies the advantages of the MVVM architecture. The separation of the architecture makes the development more modular and maintainable. For example，the technology currently used(WEB UI) is different from what was originally designed in milestone 1 (Flutter). But base on MVVM architecture, the flutter-developed client can be subsequently introduced without having to rewrite the server-side code. Flutter applications can communicate with Flask servers via HTTP requests to get data and interact with the back end. Similarly, the database can also be migrated. Flask servers can connect to MongoDB.

2.1.4 Runtime process characteristics

The key consideration I have identified for the runtime process is usability. Usability is defined as the ability for how easy a user interface can be used. There are many important aspects to the quality of a mobile application, but the most important one is usability (Nayebi et al., 2012, p. 1). Telles [1990]

states that the "interface has become an important element in garnering good reviews" of software in the trade press.

As explained earlier, I will be doing web development. so the user interface design is completely different from milestone 1. The redesigned and reimplemented UI is as follows:

- F1 User Functional runtime process

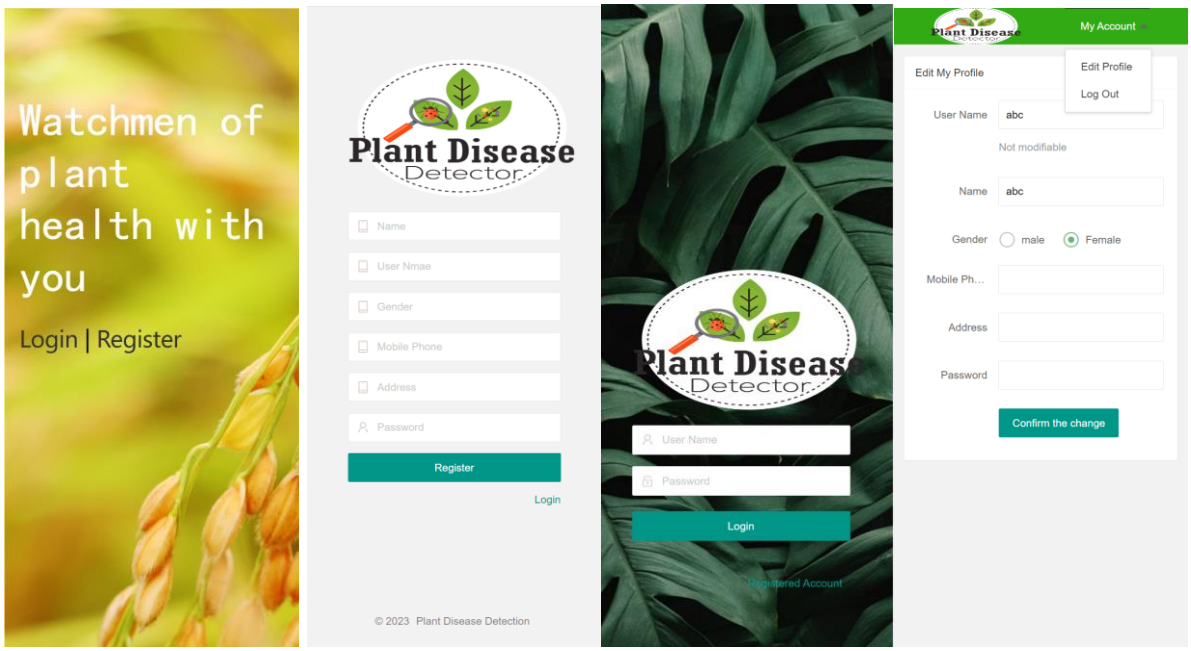

Figure 5: F1 User Functional Requirement View

Figure 5 shows the interfaces associated with F1 requirements. It mainly includes:

i. The user welcome view: where the user can choose to Login or Register.
ii. User registration view: which creates an account for a new user.
iii. User Login view: where existing users can enter the plant disease detection system.
iv. Edit Profile view: users can modify their information. Click on the icon in the upper left corner to enter the plant disease detection interface.

- F3 Disease detection functional runtime process

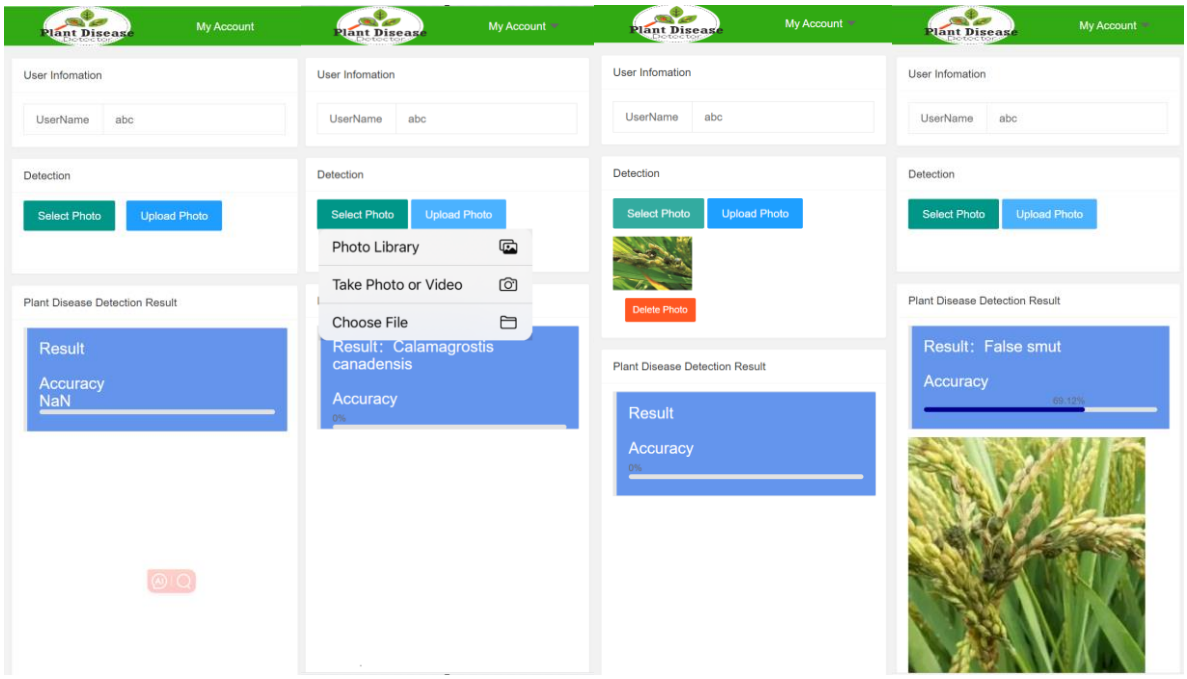

Figure 6: F3 Disease detection functional Requirement View: Unsuccessful scenarios

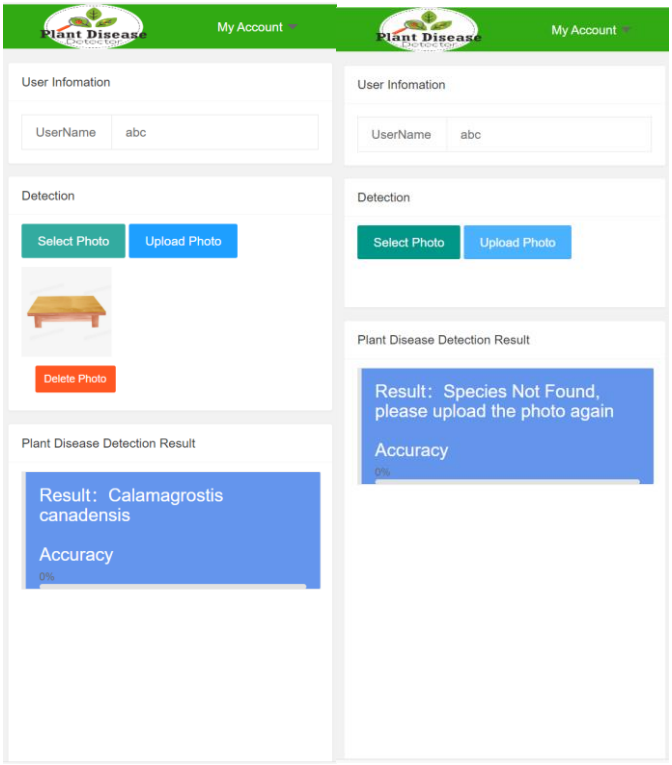

Figure 7: F3 Disease detection functional Requirement View: Unsuccessful scenarios

Figure6 and Figure7 shows the interface and operation flow of F3 requirements. The user interface flow for plant disease detection in the main interface is as follows:

i. The user selects "Select Photo", which can be a picture, a photo, or a file.
ii. The selected result will be displayed on the interface. If you want to re-select, you can select "Delete Photo".
iii. If the user wants to re-select, can select "Delete Photo".

iv. If the user wants to test plant diseases on the selected result, can select "Upload Photo".

Successful scenarios Figure6:

v. The result will be displayed below, including the name of the plant disease and the accuracy of the result.

Unsuccessful scenarios Figure7:

vi. If the image is a non-plant photo, notify the user that it is unrecognizable and please re-upload the photo.

Compared to the original design in Milestone 1, the current implementation is much more usability and users can use the system more easily. In the original sequence diagram, after the user uploads an image, an additional step is needed: to confirm whether it is the plant uploaded by the user, and only after confirming that it is the plant, the plant disease detection will be carried out. In the current operation, this step is directly omitted: after the user uploads the image, if the server confirm it is a plant, the server will directly detect the plant disease, and if it is not a plant, then it will stop the detection and notify the user, refer to Figure6 for the specific process.

2.1.5 Physical characteristics

In Milestone 1, the original design was to use Microsoft Azure as the platform for setting up the servers, mainly because its superior performance meets the high demand for feedback time of the plant disease detection system. At the same time, Microsoft Azure can flexibly adjust the Azure Load Balancer and Virtual Machine Scale Sets functions. However, in the actual development process, Microsoft Azure was not used, as a new system, in the early stages of development, there is no need to set up such a complex environment.

It is sufficient to develop and test the code in the local development environment, without immediately setting up in the cloud environment. Typically, local development environments are used for initial development, debugging, and testing, and then the application is deployed to the cloud or other production environments. This helps developers maintain independence and control over the development process and provides faster development cycles.

The database is SQLite3, and the server can interact with the database through Flask-SQLAlchemy. This is also different from the previous design in Milestone 1, which used MongoDB. The main reason for changing the database is that SQLite is more suitable for small and lightweight applications, and I am more familiar with the SQL database language. Considering the above factors, using SQLite as the database development is more maneuverable and saves development time. MongoDB can be used as an extension database in the future. MongoDB can be scaled horizontally to handle large-scale data, which is suitable for large-scale applications and distributed environments. If the project has more users in the future, it is possible to migrate the database from SQLite to MongoDB based on the existing architecture (MVVM).

2.2 Detailed description of the process and artefacts of the project

2.2.1 Development tools

Different industry procedures, instruments, and technologies were employed in the creation of this project.

- IDE: Visual Studio, PyCharm, Developer Tools
- Web UI Development: HTML, JavaScript, CSS
- Server Development: Python, Flask, PyTorch
- Services: Pl@ntNet, GitHub
- Database: Flask-SQLAlchemy, SQLite3

To improve my personal development efficiency, I test and modify my clients (WEB UI) in Developer Tools. The Device Mode feature in Developer Tools allows developers to select different mobile devices (e.g., iPhones, Android phones, tablets, etc.) and automatically adjust the size and resolution of the browser window to simulate the screen size of the selected device. Developer Tools allows me to test the responsive design of a web page to see how the page is laid out and elements are arranged on different screen sizes. This helps provide a better user experience and wider device compatibility.

In Developer Tools, I can use "Live Reload" to update the latest code in real time. This feature automatically reloads the page after I make a change to the code, allowing me to see the effect of the change immediately instead of having to manually refresh the page. This is very useful for developing and debugging web pages as it saves time and improves development efficiency.

PyCharm is a powerful integrated development environment (IDE) mainly used for Python development, which includes code editing, debugging, version control and other features. In PyCharm, I can use "Automatic Reloading" to update the code in real time. This feature allows me to automatically reload the application after making changes to the code, so that I can see the effect of the changes.

The above tools have greatly improved my development efficiency.

2.2.2 Artificial intelligence model

The plant disease detection system uses 2 deep learning models in its implementation.

One is the plant detection deep learning model, which is mainly used to detect whether the image uploaded by the user is a plant or not. The model calls the API of Pl@ntNet system to get the detection results. Pl@ntNet is a tool to help to identify plants with images. Pl@ntNet is built on the combination of a deep learning image classification model and a generalist content-based image retrieval method (Joly et al., 2014) (Affouard et al., 2017). The Pl@ntNet system includes 44102 species, 3669994 images observations and 954612676 images queries (as of Aug 2023) (Pl@ntNet. n.d.). As reported for Pl@ntNet, the correct species is returned in first position 89% of the time, and among the top 5 predicted species 97% of the time. Mean average precision is 93% (van der Velde et al., 2023).

The other is a deep learning model for plant disease detection, which is performed only after determining that the uploaded image is a plant (meet F3 and Sequence Diagram for disease detection). The model uses Residual Neural Network (ResNet) technology. ResNet is a type of deep convolutional neural network (CNN) architecture. ResNet is derived from the paper “Deep Residual Learning for Image Recognition” by Kaiming He et al. The main feature of ResNet is the use of Residual Block, which allows neural networks to avoid the “Vanishing Gradient” problem when training very deep models, making it possible to train extremely deep neural networks to advanced levels of performance (He, Zhang, Ren, & Sun, 2015).

3 System evaluation

3.1 Evaluation criteria

ISO 25010 is used for evaluating this project. The standard includes functionality suitability, reliability, performance efficiency, useability, security, compatibility, maintainability, and portability (International Organization for Standardization, 2021).

3.1.1 Functionality suitability

The evaluation criteria listed in milestone 1 provides a set of functional requirements which I will be using to test the functionality suitability of the whole system.

Use the Dimensions feature in Developer Tools to simulate the effect on your phone. Set Dimensions to iPhone 12 pro and refer to the video link for details on how to use it:

Due to time constraints, some features were therefore not implemented.

Test Case	Normal test	Abnormal test	Final Result
F1 - User Functional Test Case:			
Test case 1: Login	Given that the user is registered with the system, when they enter their username and password at the login page, then they should be able to login.	Given that the user is not registered with the system, when they enter a usernamer and password at the login page, then they should not be able to login.	Pass
Test case 2: Register/Sign Up	Given that the user is not registered with the system, when they fill out their details on the sign-up page, then they should be able to create an account and be authenticated.	Given that the user is already registered with the system, when they fill out their details on the sign-up page, they should not be able to create an account.	Pass
Test case 3: Forgot Password	Given that the user is registered with the system, when they fill out their email on the forgot password page, then they should receive an email to reset their password.	Given that the user is not registered with the system, when they fill out their email on the forgot password page, then they should not receive an email to reset their password.	This function is not implemented
Test case 4: Remove plant from My plant	Given that the user is on the my plant page, when they remove this plant from my plant, then it should be updated in the database as well.		This function is not implemented
Test case 5: Updating user profile	Given that the user is on the profile page, when they update their email, name, or password, then it should be updated in the database		Pass

	as well.		
F2 - Search Functional Test Case:			
Test case 6: Search for plant	Given that the user is on the search page, when they enter a plant name, then all information associated with that plant should show up on the screen	Given that the user is on the search page, when they enter a non-plant name, then “Not found please try again” should show up on the screen.	This function is not implemented
Test case 7: Search for plant disease	Given that the user is on the search page, when they enter a plant disease name, then all information associated with that plant disease should show up on the screen.	Given that the user is on the search page, when they enter a non-plant disease name, then “Not found please try again” should show up on the screen.	This function is not implemented
Test case 8: Add plant to My plant	Given that the user is on the plant info page, when they add this plant to my plant, then it should be updated in the database as well. User can find the new plane in my plant page.		This function is not implemented
F3 - Disease Detection Functional Test Case:			
Test case 9: Plant disease detection	Given that the user is on the disease detection page, when users utilize authentic plant images, the system provides detection results.	Given that the user is on the disease detection page, when users use non-plant images, the system returns an inability to recognize message, please reacquire the photo again.	Pass

3.1.2 Reliability

Considering the time constraints, the models used in the ResNet algorithm are Pretrained Models provided by PyTorch. The model can be used for various computer vision tasks such as image classification, target detection, etc. Using a pretrained ResNet model can greatly simplify the process of model training because it has already been trained on large-scale image data and learned rich features.

A simple comparison test was originally planned for different ResNet models. The current system uses the resnet34 model, and an accuracy comparison test with resnet50 and resne101 was planned. Accuracy is the most used performance metric to measure the percentage of correct classifications of a model over the entire dataset. The more the model's predicted categories match the actual categories, the higher the accuracy.

However, due to time constraints, the test unfortunately could not be completed.

3.1.3 Performance

The whole system does not perform well on PERFORMANCE. The main reason is that when the user uses the plant disease detection function, sometimes it will take more than 3 seconds to receive the results from the system. The main reason is that the system uses two sets of deep learning models, one of which is the plant detection API from another system, Pl@ntNet, and the time between sending a request to Pl@ntNet and receiving the result is more than 3 seconds sometimes. This experience is not user-friendly.

The plant disease detection system utilizes built-in models, resulting in rapid response times. There is significant room for improvement in terms of performance. While Pl@ntNet is a powerful plant detection system with abundant data and highly accurate detection algorithms, the feedback response time is too long, which may lead to eventual abandonment in future use.

The direction of improvement for the algorithm is to use the built-in model instead of calling external APIs, and to adjust the algorithm so that it can detect plants and plant diseases at the same time.

3.2 Evaluation results

Degree of Functionality Realization: The primary functionality has been successfully realized, which is an important milestone. However, the secondary functionality was not completed, and further development is needed to improve the functionality of the system.

Performance Issues: There were some issues with the performance of the system, especially with the feedback time. This requires further optimization.

Maintainability: The whole system has good maintainability and expansion as the whole code is based on MVVM model.

Overall, the system has had some success, but there is room for improvement, especially in terms of performance and unfinished minor features. Continue to focus on user feedback and iterative development to continually improve the quality and functionality of the system.

4 Recommendations

4.1 Process findings:

During the whole project development process, I did all the design, development, and testing by myself, which made me learn a lot of new knowledge, and found a lot of areas that could be improved. Several of the technologies were new to me, so there were many ups and downs throughout the learning process. Unfortunately, due to the time constraints of the project, it was not possible to realize every idea I had.

The evaluation sessions of each scrum allowed me to fully review the differences between the current development and the original design. This made me realize that there are a lot of shortcomings in Milestone 1. I should have done more research in Milestone 1, taking into account the time constraints, so that I could choose the tools, services, and algorithms that would be most beneficial to the project. In this way, I could have saved time during the subsequent development process and kept the entire development timeline in line with the initial development time plan.

In milestone 1, I decided to use Flutter for app development, but I only considered the advantages of Flutter and didn't plan for how the client-side of the app would communicate with the server-side. This oversight led to the inability to implement the original design in milestone 2.

4.2 Technology findings

As a detection software, its future development direction, in addition to non-stop optimization of the algorithm to improve the accuracy, but also need to be able to do real-time detection of the system. That is, open the camera, immediately show the test results in the camera range, without the need to upload pictures to get the test results. At the same time to support multi-disease real-time detection, that is, in the camera range, all the diseases of the plant can be detected, the disease can be more than one. In summary, the future of the software aims for the multi-object real-time detection.

But the main contribution of ResNet is to solve the problem of gradient vanishing in deep neural networks. It is not very suitable for multi-target real-time detection. Therefore, YOLO (You Only Look Once) algorithm can be introduced, which comes from a research paper "You Only Look Once: Unified, Real-Time Object Detection" by Redmon et al. in 2016. The YOLO model can predict object categories and bounding box coordinates and has the advantage of fast real-time detection. (Redmon, Divvala, Girshick, & Farhadi, 2016).

ResNet and YOLO have different applications and focuses in the field of deep learning. Therefore, they can be combined, such as using ResNet for feature extraction in image classification tasks and subsequently applying YOLO for object detection on these features. Ganesan and his colleagues have conducted similar experiments and achieved higher recognition rates using Res-Yolo (Ganesan & Chinnappan, 2022).

References

Affouard, A., Goëau, H., Bonnet, P., Lombardo, J.-C., & Joly, A. (2017). Pl@ntNet app in the era of deep learning. In International Conference on Learning Representations (ICLR), Toulon, France.

Aggarwal, V., & Singhal, A. (2019). Empirical Study of Test Driven Development with Scrum. In M. Singh, P. Gupta, V. Tyagi, J. Flusser, T. Ören, & R. Kashyap (Eds.), Advances in Computing and Data Sciences. ICACDS 2019. Communications in Computer and Information Science, vol 1046. Springer, Singapore. https://doi.org/10.1007/978-981-13-9942-8_2

Brownlee, J. (2016). *Deep Learning With Python: Develop Deep Learning Models on Theano and TensorFlow Using Keras*. Machine Learning Mastery.

Copperwaite, M., & Leifer, C. (2015). Learning Flask Framework. Packt Publishing Ltd

Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. Computers and Electronics in Agriculture, 145, 311-318. Elsevier.

Ganesan, G., & Chinnappan, J. (2022). Hybridization of ResNet with YOLO classifier for automated paddy leaf disease recognition: An optimized model. *Journal of Field Robotics, 39*(7), 1087-1111. https://doi.org/10.1002/rob.22089

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. Microsoft Research. https://arxiv.org/pdf/1512.03385.pdf

International Organization for Standardization. (2011). Systems and software engineering Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models (ISO Standard No. 25010:2011). https://www.iso.org/standard/35733.html

Joly, A., Goëau, H., Bonnet, P., Bakić, V., Barbe, J., Selmi, S., ... Barthélémy, D. (2014). Interactive plant identification based on social image data. Ecological Informatics, 23, 22-34. https://doi.org/10.1016/j.ecoinf.2013.07.006

Memon, M. S., Kumar, P., & Iqbal, R. (2022). Meta Deep Learn Leaf Disease Identification Model for Cotton Crop. Computers, 11(7), 102. https://doi.org/10.3390/computers11070102

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in Plant Science, 7, 1419.

Padol, P. B., & Yadav, A. A. (2016). SVM Classifier Based Grape Leaf Disease Detection. In IEEE Conference on Advances in Signal Processing (CASP), Pune (pp. 175-179).

Pantazi, X., Moshou, D., & Tamouridou, A. (2018). Automated leaf disease detection in different crop species through image features analysis and One Class Classifiers. Computers and Electronics in Agriculture, 156, 96–104. [Google Scholar] [CrossRef]

Pawar, R., & Jadhav, A. (2017). Pomegranate Disease Detection and Classification. In IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai (pp. 2475-2479).

Pl@ntNet. (n.d.). Pl@ntNet: An Image-Based Plant Identification Application. https://identify.plantnet.org/

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *arXiv:1506.02640 [cs.CV]*. https://doi.org/10.48550/arXiv.1506.02640

Shruthi, U., Nagaveni, V., & Raghavendra, B. K. (2019). A Review on Machine Learning Classification Techniques for Plant Disease Detection. In 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS). CSE Department, AcIT, Bengaluru, India.

van der Velde, M., Goeau, H., Bonnet, P., d'Andrimont, R., Yordanov, M., Affouard, A., Claverie, M., Czucz, B., Elvekjaer, N., Martinez-Sanchez, L., Rotllan-Puig, X., Sima, A., Verhegghen, A., & Joly, A. (2023). Pl@ntNet Crops: Merging citizen science observations and structured survey data to improve crop recognition for agri-food-environment applications. Environmental Research Letters, 18(2), 025005. https://doi.org/10.1088/1748-9326/acadf3

Yahya, N., & Maidin, S. S. (2022). The Waterfall Model with Agile Scrum as the Hybrid Agile Model for the Software Engineering Team. In 2022 10th International Conference on Cyber and IT Service Management (CITSM) (pp. 1-5). Yogyakarta, Indonesia. doi:10.1109/CITSM56380.2022.9936036.