Plant Disease Detection Mobile System

COMP826 Mobile Systems Development 2023 S2
Milestone 1: Scoping Study and System Design

Prof. Roopak Sinha
Dr. Matthew Kuo

WULAN E

22169691

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

Sep 01, 2023

Table of Contents

1 Research Topic, Scope, and Role

Plants are important part of human life, thriving in every corner of our existence. The detection of plant diseases is not just a scientific endeavor; it is a responsibility to safeguard ecological equilibrium. The identification of plant diseases has a wide range of practical application value in the field of agriculture.

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 Project Description and 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 Scope and Role

1.2.1 Key Requirements

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.

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. The best is My Plant function enables users to access previous identification records at any time.

Complex Features of the Plant Disease Detection is the disease database and detection algorithm. Achieving high accuracy in image recognition and analysis demands intricate computer vision algorithms and Deep Learning algorithms. These algorithms require training on extensive image data and must adapt to diverse plant and disease types. And plant diseases come in diverse forms, and similar symptoms may appear in different diseases or even exhibit variations in the same disease across different plants. Accurate identification and classification are thus complex endeavors. Developing and maintaining a comprehensive database with extensive plant disease information is challenging. Accurate disease identification and recommendations rely on up-to-date and validated database content.

1.2.2 Development Scope and Role

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). This document does not contain Image recognition algorithm design, Disease detection algorithm design and Actual image training.

2 Development Process

2.1 Development processes used in industry

2.1.1 Waterfall

The waterfall model can be categorized as a traditional Software Development Lifecycle (SDLC) and is known for its linear fashion phase development. The waterfall model comprises six (6) phases: analysis, design, development, testing, implementation and maintenance. Figure 1 depicts the waterfall model. The waterfall model remains one of the preferred models by a software engineering team. The preference is suggested due to the benefits of the waterfall model as such: easier to manage, the deliverable and milestones are well-defined before the project starts, the project initiation and planning were adequately constructed, and all the phases and its activities are outlined. In addition, the Waterfall model works well on big and weak teams (Rahim et al. 2018). Theocharis et al. (Theocharis et al. 2015) reported that project managers have the responsibility to align projects with the respective company strategy, such as projects must interface further the business processes like human resource management, sales, contracting, and so forth and that needs is one reason to answer the question on why Waterfall is still relevant in today software development project (Yahya and Maidin 2022).

2.1.2 Scrum

Scrum was first introduced in 1997 (Schwaber 1997) and has since become the most widely applied agile software development framework (VersionOne, 2017). At its core, Scrum splits development into iterations not longer than four weeks (called sprints). At the end of each sprint, a shippable product increment is delivered to the user. For each new sprint, a sprint-planning meeting is held, at which tasks for the sprint are selected by the developers themselves in collaboration with other stakeholders.

In Scrum, the customer is represented in the role of the product owner. Requirements are captured in the form of user stories and are aggregated in a prioritized product backlog. The product backlog is a "living" document, as it is updated continuously and thus reflecting the current understanding of user needs (Hron and Obwegeser 2018).

2.1.3 Hybrid Agile Model

A combination of development approaches known as hybrid agile has started taking interest among the software engineering team. Hybrid agile combines a plan-driven development model and an agile process model in one software project. The plan-driven and agile models are combined to best fit a project's needs, and the combination offers a wide range of advantages, such as it improves the project quality, a project being delivered on time, and enhancing the accuracy of resources, especially the workload estimation. In addition, hybrid agile focuses on the user's need rather than process and continuous product improvement iteratively. Among models that have been adopted that create hybrid agile is the combination of Scrum and Waterfall, Scrum, eXtreme Programming, and Lean Software Development , and Scrum and eXtreme programming (XP). The combination of development models is due to the strengths and shortcomings of distinct approaches; a combination of the two methods has been proposed as a feasible solution for overcoming one method's weakness by replacing it with the strengths of the other development model (Yahya and Maidin 2022).

2.2 Selection of development process

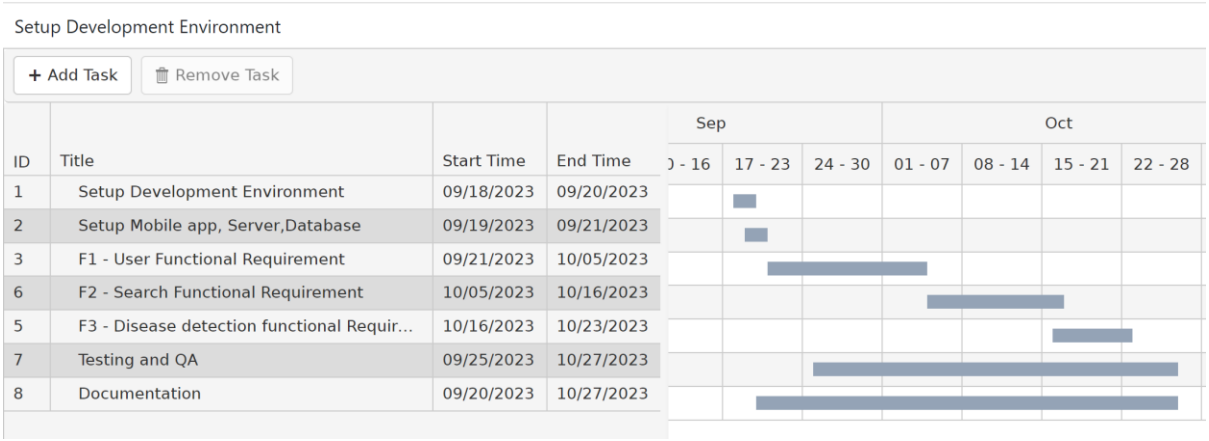
The development process I have chosen for this project is a Hybrid Agile Model which is combination of Scrum and Waterfall.

Considering the limited time available and the fact that I am the only developer, all the requirements (F1, F2, F3) will be defined upfront and will not change later, and the final release date of the development will not change. This aligns with the strengths of the Waterfall model. The Waterfall is regarded as the preferable model because of its strength in ensuring the stability of requirements before starting a software project. In Waterfall, the requirements are clearly outlined before the development phase, and there is a confirmation that the requirement will not have any changes, major changes, or frequent changes, and projects are with a significant number of tasks (Yahya and Maidin 2022).

However, during the specific code development phase, I adopt the Scrum model. As software intended for use on mobile clients, the primary objective is customer-centric development. Therefore, continuous communication with customers, along with iterative modifications and testing, is essential throughout the development process. This demands a flexible development approach, a characteristic well-met by Scrum. Scrum attracts the practitioner's interest to adopt because the technique promotes active communication with the product owner and stakeholders, leading to the project being developed, aligned, and meeting the goal (Yahya and Maidin 2022).

2.3 Plan and Gantt Chart

Figure 1: Development Gantt Chart



The Gantt chart shown in Figure 1 above displays the developments that must be completed within 6 weeks. The designation of the development plan is mainly based on the previous features, including F1, F2, F3. However, the specific phases of development, testing, and documentation follow the Scrum model. Each feature serves as a sprint, aligning with F1, F2, and F3 to generate three distinct sprints. Each sprint entails development, testing, modifications, and documentation tasks. Upon completion of each sprint, an independent functional requirement is generated. However, each sprint's development is based on the progress made in the previous sprint. After all development is complete, there is an integration test and performance test at the end.

3 Technology

3.1 Technology used in industry

The plant disease detection system needs to use the iOS and Android systems, which account for more than 99.9% of the mobile device operating systems market. For cross-platform solutions, we selected the three most popular ones: Xamarin, Flutter, and React Native, developed by Microsoft, Google, and Facebook, respectively.

3.1.1 Xamarin

Xamarin Framework helps you to build a crossplatform mobile application by offering a single language using C#, class library, and runtime that works across all major mobile platforms of iOS, Android, and Windows Phone. It combines all the power of native platforms and adds a number of powerful features of its own. Using C# language, API and Data structures on an average you can share 70% of app code cross all major platforms and with Xamarin.Forms building user interfaces you can share approximately 100% (Prajapati, Phadake, & Poddar, 2016).

3.1.2 React Native

(React Native) was developed by Facebook and released in 2015. It is based on the React framework used for web development, but it enables the use of native UI elements in mobile applications, unlike websites or frameworks like Apache Cordova. Much of the application was written using JavaScript, optionally, in TypeScript. Native UI elements and services like Bluetooth, cameras, and sensors are run using native code (Nawrocki et al., 2021).

3.1.3 Flutter

Flutter, the third framework, was released in 2019 and allows for the development of mobile applications for Android and iOS. This cross-platform solution was implemented in C/C++, Dart, and

uses Google's Skia Graphics Engine. Unlike Xamarin and React Native, Flutter does not use native UI elements; instead, they are drawn using the Skia Graphics Engine. Flutter allows the use of "stateful hot reload"; changes are displayed in real time without rebuilding the entire application (Nawrocki et al., 2021).

3.2 Technology Choice

According to the data from statista.com, React Native 29%and Flutter 46%are currently the two most widely used frameworks in 2022. Considering that my system requirement F3 involves a detection system with strong performance demands, Xamarin falls short of meeting my criteria. Therefore, I have chosen to abandon its use. Nawrocki et al. (2021) mention that Xamarin has problems in very important areas; most noticeably, the start-up time on Android is a big detriment to user experience. Even for a "Hello World" application, the start-up time exceeded 2 s, and a more advanced application took nearly 3 s to open. Another big issue is application size. On top of that, Xamarin does not have any hot reload tools built in, significantly slowing down its development (Nawrocki et al., 2021).

Next main comparison React Native and Flutter.
The length of the feedback cycle is a very important aspect of mobile application development. Especially when designing the user interface, long feedback loops can significantly increase development time and decrease developer satisfaction. Of all the frameworks compared, Both React Native and Flutter provide hot reload tools for free. During the development of benchmark applications, Flutter's hot reload solutions proved to be slightly faster and much more stable, as React Native sometimes did not refresh changes or refreshed the changed code twice (Nawrocki et al., 2021).

The important characteristic of the framework is how easy it is to use the tools available. In this aspect, of all the cross-platform frameworks, Flutter seemed to have the best quality tools. For the initial environment setup, compilation, deployment, and profiling, Flutter provides easy-to-use tools that work well. The same cannot be said of the other frameworks. The React Native compilation for release builds can take a very long time, and the time of completion is quite indeterministic. Sometimes the release build was completed in 2 min, and in other cases, 10 min were required. Such indeterminism was not noticed for any other framework (Nawrocki et al., 2021).

Combine the above comparative information, the technology I have decided to choose for this project is Flutter. Since I lack prior experience in frontend development, I'll need a framework is easy use. And considering requirement F3 has high performance requirements and high feedback time requirements. Therefore in terms of deployment, Flutter provides a more user-friendly approach for me.

3.3 Technology Configurations

- Programming Languages: Typescript, Python
- Server: Microsoft Azure Server
- Version control: Github
- Testing: Appium
- Track issue/feature: Trello
- Framework: Flutter,
- Database: DBMS MongoDB Cloud
- IDE: VS Code, Android Studio, Xcode

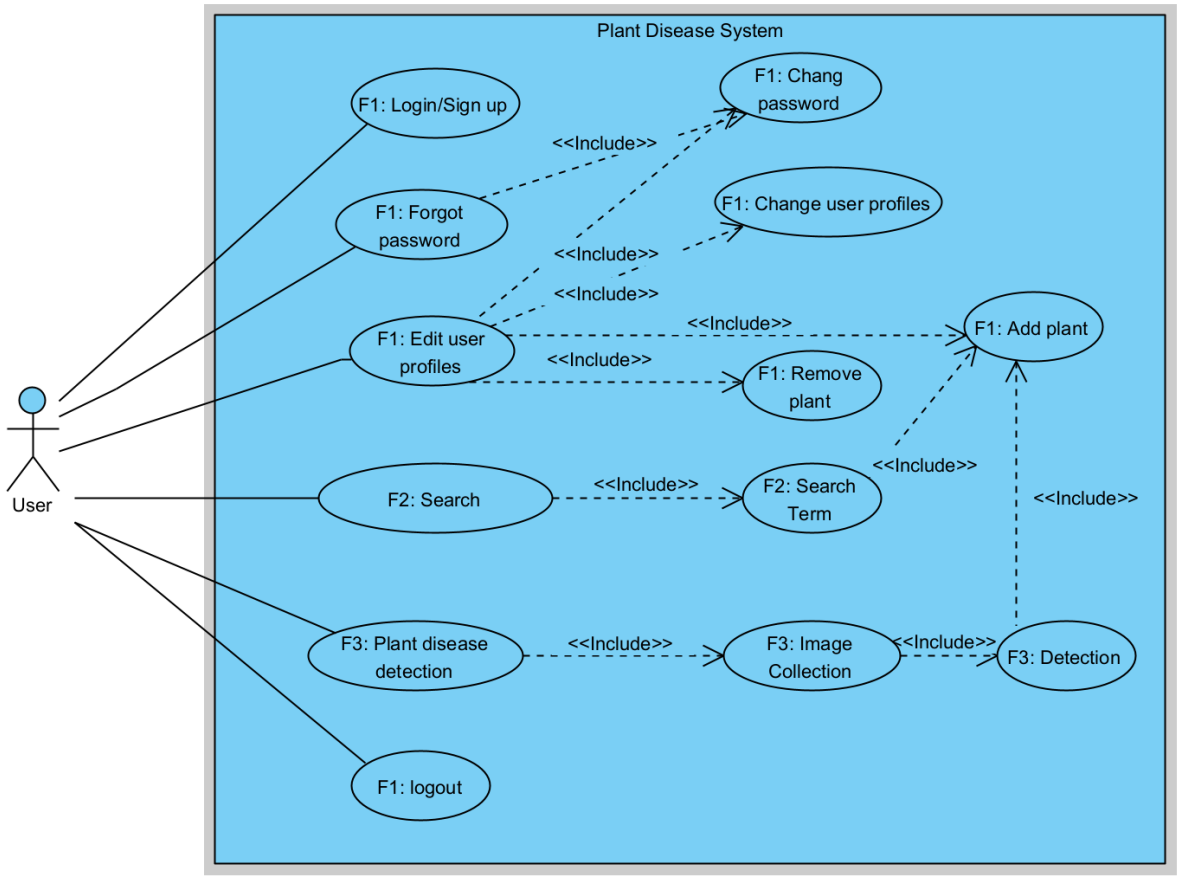
4 Design of Deliverables

4.1 Logical Decomposition

4.1.1 Key Considerations

According to the features (F1, F2, F3) described above, list more detailed decomposition from the user's point of view:

Figure2: User Case Diagram



As shown in Figure 2 above, the main potential aims of the user for using the system.

F1 User Functional Requirement include:

- User login/logout: User identity authentication and logout system.
- Forgot/Change password: Helps users authenticate and reset passwords.
- Edit User account: Allow users to change some user information, such as username, password, etc.
- Edit My Plant: “My Plant” is also a part of the user account. Users can add or remove plant to or from the “My Plant”. “My Plant” enables users to access previous identification records at any time, aiding in monitoring plant health and serving as a reference for determining the effectiveness of treatment measures.

F2 Search Functional Requirement include:

- Search plant or plant disease: The user enters the search view, enters the name of the plant or plant disease in the search term, and gets the corresponding search result. Users can also choose whether to add the Plant to "My Plant" based on the query results.

F3 - Disease detection functional Requirement include:

- Detect plant disease: The user enters the disease detection view; the user needs to collect plant pictures correctly according to the requirements. After the pictures are collected, the user can click "Detection" for disease detection. The system will return the detection result.

4.1.2 Quality attributes scenario

QA scenarios are gathered through QA workshops. There are three QA scenarios listed below, which cover use case scenarios, growth scenarios and exploratory scenarios.

scenario 1	Users can easily utilize the plant disease detection feature.
QA 1	Usability: User Error Protection
Stimulus	Users collect and upload plant images or non-plant images.
Response	The system operates smoothly.
Response measure	When users utilize authentic plant images, the system provides detection results. If users use non-plant images, the system returns an inability to recognize message, please reacquire the photo again.

scenario 2	Users can swiftly obtain detection results for plants or plant diseases.
QA 2	Performance Efficiency
Stimulus	The number of requests from application users is rapidly increasing over time.
Response	The system operates smoothly and provides results promptly.
Response measure	The average response time should be 2 seconds, with a response time not exceeding 3 seconds at peak times.

scenario 3	Users receive detection results with high accuracy and precision.
QA 3	Functional Correctness
Stimulus	Users correctly obtain three images of plant diseases
Response	the system provides the top matching detection results for user selection.
Response measure	At least one of the returned results has a matching rate exceeding 80%.

Quality attributes (QA) defined in the ISO/IEC 25010 standard. Three diagrams were included in the following report, these diagrams are Class, Sequence, and Deployment diagrams. These diagrams cover the Usability, functional Correctness, and performance.

4.1.3 Choice of Architectural Pattern

According to the research done by Nunkesser (2021), there are two main goals when choosing an architectural pattern: independence, each code section should have a single purpose with the ability to be reused in other programs and unit tested. This offers better flexibility and maintainability. Automatic testing is the second factor where testing can be divided into three categories: local unit tests (on the physical hardware), unit tests (on the runtime/emulated hardware), and UI tests. The

benefit of testing provides the developer with instant feedback and prevents problems to occur when the product is deployed to the customers.

By comparing the evaluation of three different design patterns MVC, MVP, and MVVM research papers, I can conclude that 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. These advantages are in line with Nunkesser's (2021) two main goals for choosing design patterns.

4.1.4 Details of Logical Decomposition

Designing a plant disease detection app using the MVVM pattern:

F1 - User Functional MVVM:

- View Layer: User-friendly Interface: Create an interface that allows users to easily upload images and provides feedback to let users know their actions are being processed. Meet QA1 Usability
- ViewModel Layer: User Interaction Logic: Implement logic in the ViewModel to ensure users can easily upload images and interact with the application, such as clicking buttons or viewing results.
- Model Layer: User model

F2 - Search Functional MVVM:

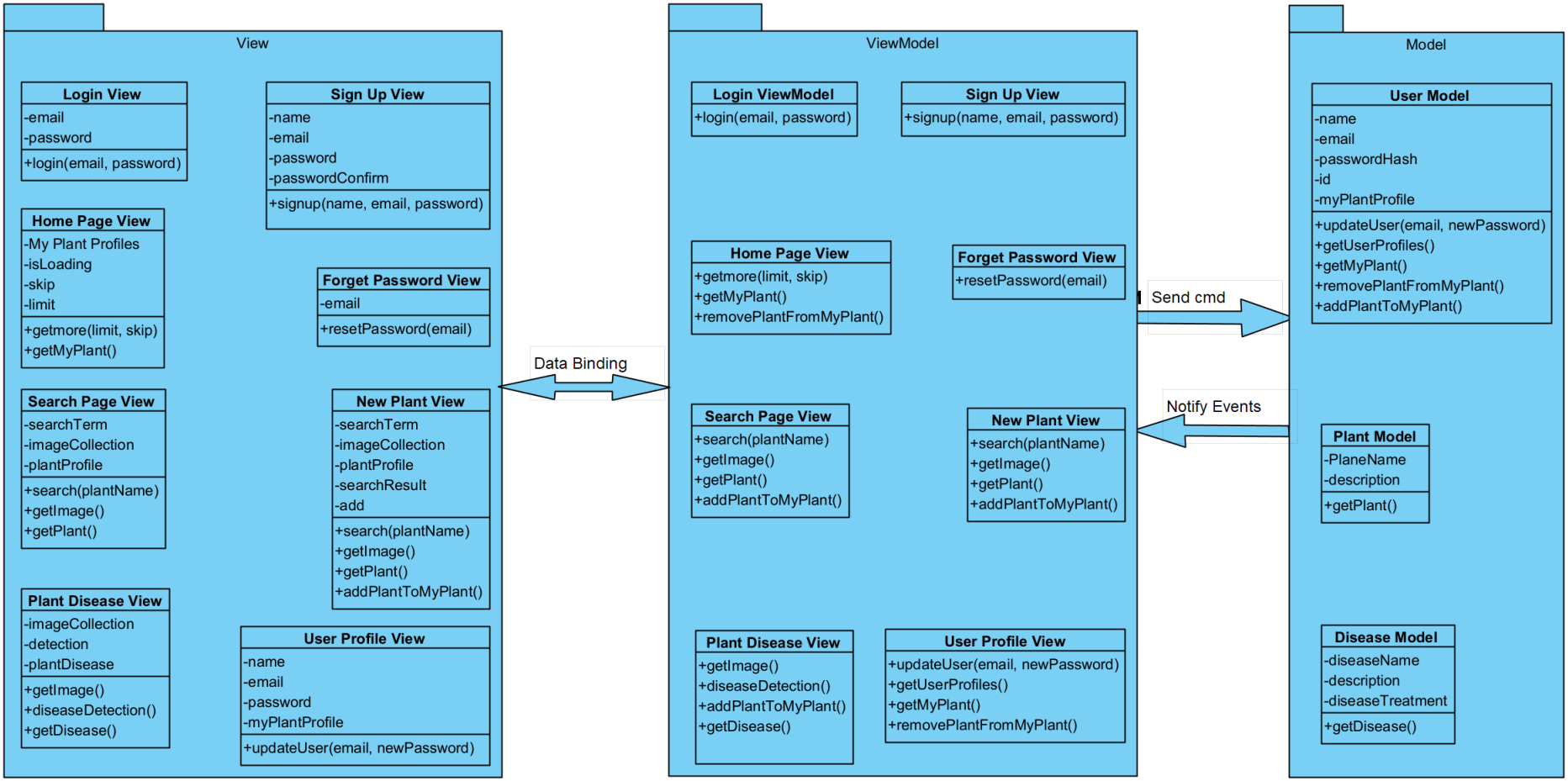
- View Layer: Design Search Interface: Develop a search interface where users can input relevant plant name or disease name to conduct searches.
- ViewModel Layer: Search Logic: Implement search logic in the ViewModel, passing user input to the Model layer for searching and returning search results to the View layer for display.
- Model Layer: Plant model and Disease Model

F3 - Disease Detection Functional MVVM:

- View Layer: Image Upload Interface: Design an interface that allows users to upload plant images for disease detection. Result Presentation: Display disease detection results on the interface for users to understand the outcome.
- ViewModel Layer: Image Processing Logic: Implement image processing algorithms in the ViewModel to extract necessary features from uploaded images for disease detection. Disease Detection Logic: Utilize the ViewModel to process image features, run disease detection algorithms, and return detection results to the View layer. In the ViewModel layer, optimize the image processing algorithms to ensure swift and accurate disease detection while maintaining the application's smooth performance. Meet QA2.
- Model Layer: Disease Model

Class Diagram:

Figure3: MVVM Class Diagram



As shown in Figure 3 above, The MVVM class diagram has been modeled to fit the Features of this mobile system application. Views contain user interface elements and business logic. The Viewmodel connects the View to the Model by taking actions from the View and updating the Model; likewise, the Model notifies the Viewmodel of data changes and updates the View accordingly. The Model maintains the data and the connection to the database, allowing the data to be updated and queried. This design allows new features or changes to be added to the code without too much difficulty. This makes the mobile system application maintainable.

4.2 Runtime Process Characteristics

4.2.1 Key Considerations

The key consideration I have identified for the runtime process is QA1 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.

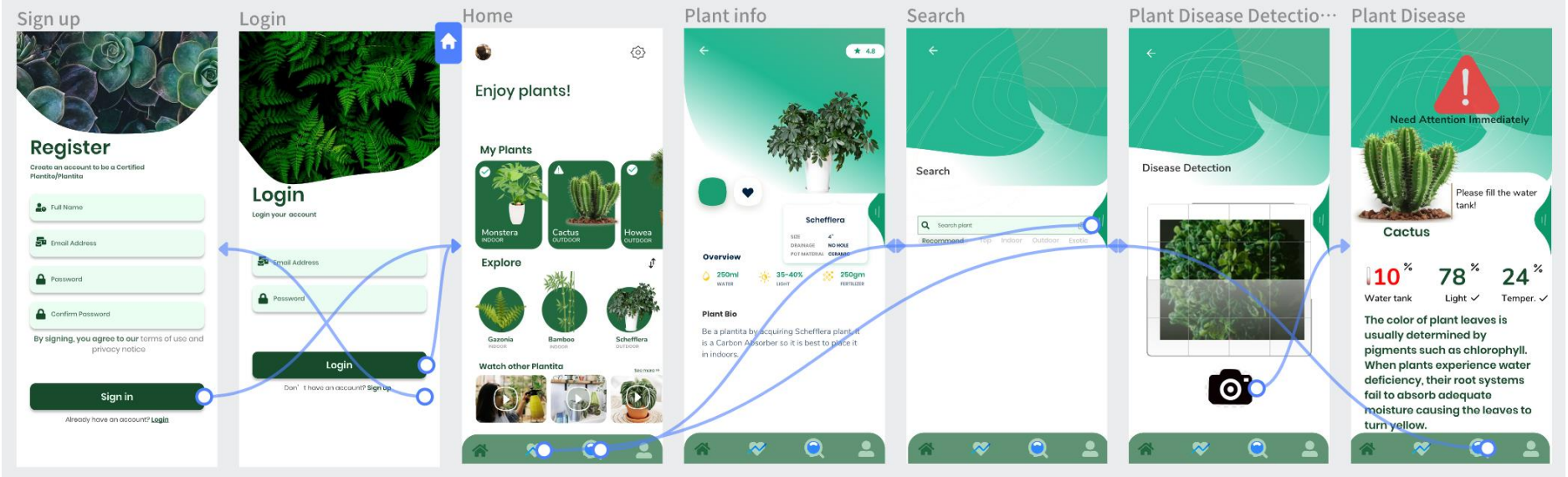
4.2.2 User Interface

Many companies around the world have created their own design system as an effort to set their brand apart from the competition. Unique design systems allow the brand to maintain a good reputability. Design system is a collection of user interface resources that a company have. These varies from documentation, code snippet, image examples, tools, and design guidelines. Each component is designed to be reusable and repeatable (UXPin, 2021). The top design system used by developers and companies around the world are Google's Material Design (29%), Apple's Human-Interface Guidelines (15%), and Bootstrap (10%) (Hamilton, 2020). This shows that the Google's Material Design system is the most popular around the world, as a result, I will be using this design pattern when designing the user interface of this mobile application.

Nielsen, J. (1993) categories usability into five principles: Easy to learn, Efficient to use, Easy to remember, Few error, Subjectively pleasing. The user interface screens will be designed based on the five principles. The following images include the main user interface interfaces, but do not list all of them.

Combine the above 2 points, each UI page will be designed using Google's Material Design system as it is the most commonly used design system around the world. This is beneficial for providing a likeable/familiar interface for users using this mobile application for the first time, as the user's knowledge and experience from other Material design apps can be transferred and applied in this new mobile application. Meanwhile, each UI page will be designed according to Nielsen principles.

Figure 4: UI and Navigation Diagram



As shown in Figure 5, the navigation lines are also drawn to demonstrate how different actions on the screen will take you to another screen. The interface provides a familiar design to other website and mobile applications, this effectively reduces the amount of time the user needs to spend learning the new mobile application. The design is similar to other application the user has used, then it will allow the user to learn the interface easily. Meet QA1.

F1 - User Functional page: Sign Up page, Login page, Home page. My Plant function in the home page. After successful login, the user will enter Home page.

F2 - Search Functional page: Search page, Plant info page. User input the plant name in the Search page, then the app display Plant info page which is the result of search.

F3 - Disease Detection Functional page: Plant Disease Detection page, Plant Disease page. User collect the plant images in Plant Disease Detection page, then the app display the Plant Disease page which is the detection result.

4.2.3 Details of Runtime Process

The ISO 25010 standard states that usability is the degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use. The ISO 25023 standard categories usability measure into six sections: Appropriateness recognizability measures, Learnability measures, Operability measures, User error protection measures, User interface aesthetics measures, Accessibility measures.

User error protection measures are used to assess the degree to which the system protects users against making errors. NOTE User error protection measures are expected to be measured through operational testing by representatives of operators or end users or can be measured through static analysis such as review of requirement, design specification or user manuals.

Interpret F3 through sequence diagram.

Figure 5: Sequence Diagram for disease detection

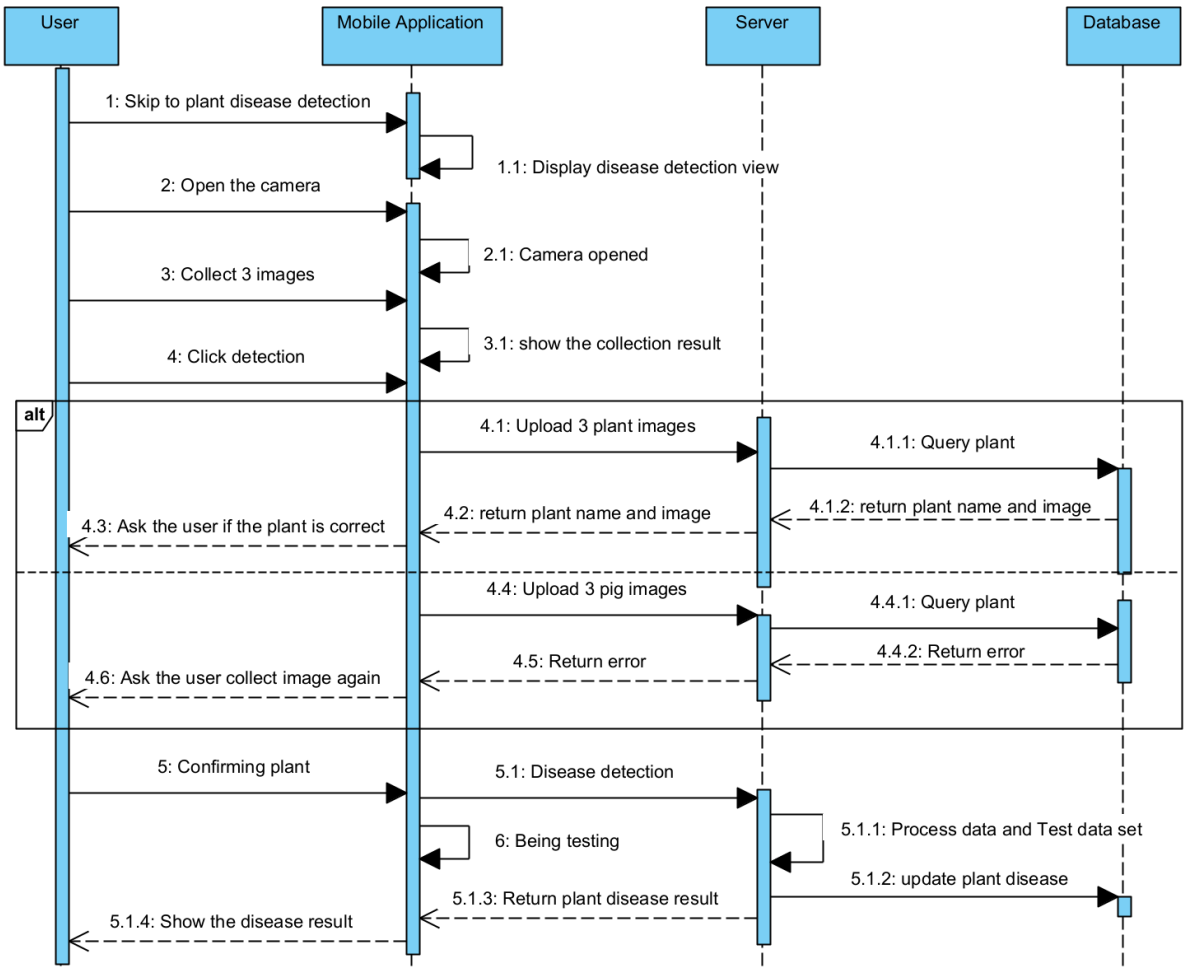


Figure 5 is derived from the detailed process for F3 and meet QA1. It fully demonstrates the plant detection feature in a plant detection application, including both successful and unsuccessful scenarios. The unsuccessful scenarios proves that the system support QA1.

- The user enters the plant disease detection page.
- The user selects or collect three images of a plant as input images.

- The application receives the input image and sends it to the server for processing.
- The server receives the image and attempts to use image recognition algorithms for plant detection.
- successful scenarios:
    i The server retrieves information about the plant from the database, such as its name, features, and diseases.
    ii The server sends the plant detection results and relevant information back to the application.
    iii The application displays the detection results and relevant information to the user.
    iv The user can view the recognized plant's name, features, and potential disease information.

- unsuccessful scenarios:
    i The server encounters issues during the image recognition process, unable to accurately identify the plant disease.
    ii The server sends the error detection results to the application.
    iii The application displays the error results and suggest the following operations.
    iv The user can choose to collect three new images or try other solutions.

4.3 Physical Characteristics

4.3.1 Key Considerations

Server architecture diagrams are often presented in a physical view, which is used to describe the mapping relationship between system software and physical hardware. It reflects how system components are deployed onto a group of computational nodes and guides the deployment implementation process. The primary audience for these diagrams is typically operations and implementation personnel. In reality, the setup of server architecture varies based on factors such as business scenarios, data volume, user load, etc. There is no fixed deployment plan; it follows a process of weighing pros and cons. After comprehensive consideration, the optimal approach is chosen.

The key consideration I have identified for the Physical is QA2: Performance Efficiency.

According to ISO 25023, performance efficiency measures are used to assess the performance relative to the amount of resources used under stated conditions. Resources can include other software products, the software and hardware configuration of the system, and materials (e.g. storage media). The performance efficiency measure is affected strongly and fluctuates depending on the conditions of use, such as load of processing data, frequency of use, number of connecting sites and so on.

As a client-server model system, the performance of server is very important. Considering that the disease detection system is a completely new product, users will grow over time. Purchasing new hardware servers would not only incur high costs but also require dedicated personnel for server maintenance, leading to additional expenses and manpower. Therefore, I have opted for cloud servers Azure. Cloud servers Provides flexible scalability. It can dynamically adjust server resources based on user traffic changes. Its performance is shown as follows:

Within the NIST definition of cloud computing, three service models exist: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). IaaS is defined as the consumer's ability to provision processing, storage, networks, and other fundamental computing resources, where the consumer can deploy and run arbitrary software, which includes operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; Each of the Azure compute services offer various scalability and service-level agreements (SLAs), ranging from 99% to 99.99%. Don't worry about having all your cloud resources or services in the same Azure region. While some workloads may need this, traversing the Azure global network to get the service you need is not an issue thanks to high throughput and extremely low latency. For example, in the United States, a connection from the West Coast to the East Coast can be made in less than 60 ms (milliseconds). From Colorado, a user can connect to the East Coast in less than 45 ms (Soh et al., 2020).

From the above content, it is evident that Azure aligns with the requirements of QA2. Considering the increasing number of users in the future, the introduction of load balancing is necessary. According to Akamai Technologies (n.d.), cloud load balancing is the practice of evenly distributing traffic, workloads, and client requests across multiple servers running in a cloud environment. This practice delivers superior cloud optimization by ensuring that each cloud resource has a load it can reasonably manage, preventing machines or servers in a cloud environment from being either overloaded or underutilized. Effective load balancing allows organizations to meet the demands of cloud-based workloads while improving performance, enhancing reliability, minimizing downtime, and reducing latency (Akamai Technologies n.d.).

In summary, we need to utilize Azure's IaaS, Azure Load Balancer, and Virtual Machine Scale Sets.

Another key consideration is QA3: Functional Correctness.

According to ISO 25010, Functional Correctness means degree to which a product or system provides the correct results with the needed degree of precision.

In order to provide users with an enhanced user experience, it is imperative to ensure that users receive accurate detection results which is consistent with QA3. Not relying solely on advanced deep learning algorithms and convolutional neural networks, but also relying with employing large-scale, high-quality plant images and disease sample datasets for model training. Therefore, the database is very important in the whole system. The database must not only be capable of storing large datasets but also possess robust disaster recovery and backup mechanisms.

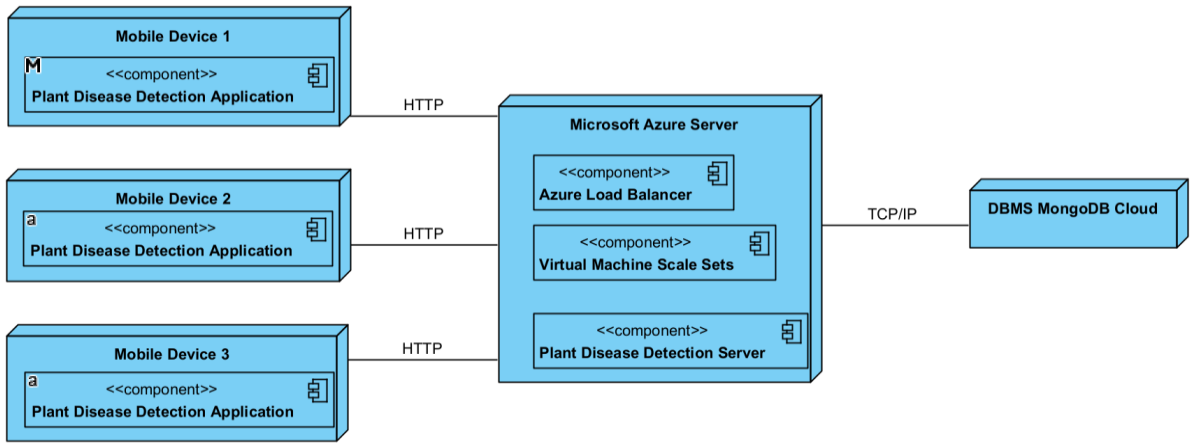
For the same reason that I choose Server, I have opted for cloud storage DBMS MongoDB Cloud.

Now a days with the demand of increasing data, NoSQL databases got the importance. Relational database is not capable of handling large data sets and even it fails to store unstructured data. Biggest advantage of NoSQL database is, that they are schema less, so any kind of data format is supported. While in SQL you have to stick to specific format for storing data. Scalability is an important issue and you can store structured, unstructured or semi structured data in it. Even NoSQL is major support for wide adoption of Cloud computing. Currently widely used NoSQL databases are MongoDB, Cassandra, DynamoDB, CouchBase etc (Gandini et al., 2014).

For image dataset storage I have selected MongoDB. The purpose behind selecting the NoSQL databases is that it falls under same category of Document based. Among the popular NoSQL databases mentioned above, MongoDb has shown good read, write and delete performances (Li & Manoharan, 2013).

4.3.2 Overall System Configuration

Figure 6: Deployment Diagram



Mobile Device:

- Refers to a category of electronic devices that are portable and designed for various tasks including communication, entertainment, work, and more. These devices are characterized by their small size, lightweight nature, portability, and ease of carrying, allowing users to perform a wide range of activities regardless of their location. Such as Smartphones, Tablets.
- Users can operate the system from their mobile devices. Users can download and install the Plant Disease Detection application from the app store. Once the application is opened, they can follow the instructions to operate F1, F2, F3.
- Mobile devices typically communicate with Microsoft Azure servers through HTTP/HTTPS networking protocols and technologies.
- Mobile application: It is a user interface includes user profile creation, modification, and login. It involves capturing plant photos using the camera and sending the photos to the server for plant or plant disease detection. It also involves recording and maintaining plant information specific for each user.

Microsoft Azure Server:

- Microsoft Azure is an overarching brand name for Microsoft's cloud-computing services. It covers a broad, and still growing, range of services that often form the foundational elements of cloud computing (Copeland, Soh, Puca, & Microsoft Azure, 2015).
- In the plant disease detection system, it contains three important functions:
  i Azure Load Balancer: Using Azure Load Balancer on Azure ensures load balancing by evenly distributing requests across multiple virtual machines.

ii Virtual Machine Scale Sets: Using Azure Load Balancer on Azure ensures load balancing by evenly distributing requests across multiple virtual machines.

iii Plant Disease Detection Server: It forms a client-server model with the "Plant Disease Detection Application." Through this model, users can operate F1, F2, and F3 using mobile devices and send corresponding commands to the server. The server is responsible for executing instructions related to F1, F2, and F3 then returning the execution results back to the application. It also needs interacting and synchronizing data with the database. Utilizing deep learning techniques and computer vision algorithms to automatically identify plants or plant diseases.

DBMS MongoDB Cloud:

- MongoDB cloud services consist of a comprehensive suite of data products that accelerate and simplify how you build with data for any application. With Atlas Database (the Database-as-a-Service for MongoDB), Search, and Data Federation, you can serve any class of workload through a common API (MongoDB. n.d.).

  i Database: Storing and managing user data, application configurations, user settings, plant data, plant disease data, and other related information. Servers are responsible for managing these data, ensuring their security, consistency, and availability.

5 Evaluation Criteria

5.1 Key Considerations

Evaluation can be defined as the process to assess the accomplishment of technical and operation effectiveness (AcqNotes, 2017). Key factors to consider when planning evaluation criteria are clear, realistic, measurable, relative, and justifiable. The criteria must be clear and not ambiguous, the criteria should be realistic to the nature of the context, the criteria should be measurable and have an importance, key factors of the project requirements must relate to the evaluation criteria, and the criteria just be justified on a technical basis (AcqNotes, 2017).

5.2 Detailed Evaluation Criteria

According to the F1, F2, and F3, each sprint entails the relating testing. After all development is complete, need to do all the test again(integration test) at the end. Finally, after finished all integration test, the performance testing is required to meet the QA2: The average response time should be 2 seconds, with a response time not exceeding 3 seconds at peak times.

F1 - User Functional Test Case:

Test case 1: Login

- Given that the user is registered with the system, when they enter their email 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 an email and password at the login page, then they should not be able to login. Meet QA1 User Error Protection testing

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. Meet QA1 User Error Protection testing

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. Meet QA1 User Error Protection testing

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.

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 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. Meet QA1 User Error Protection testing

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. Meet QA1 User Error Protection testing

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.

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. Meet QA3: At least one of the returned results has a matching rate exceeding 80%.

- 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. Meet QA1 User Error Protection testing.

References

AcqNotes. (2017, July 27). Evaluation criteria. https://acqnotes.com/acqnote/tasks/evaluation-criteria

Agile Alliance. (n.d.). Manifesto for Agile Software Development. https://agilemanifesto.org/

Akamai Technologies. (n.d.). Cloud Load Balancing. Retrieved from https://www.akamai.com/glossary/what-is-cloud-load-balancing

Flutter. (n.d.). Beautiful native apps in record time. https://flutter.dev/

Gandini, A., Gribaudo, M., Knottenbelt, W. J., Osman, R., & Piazzolla, P. (2014). Performance Evaluation of NoSQL Databases. In A. Horváth & K. Wolter (Eds.), Computer Performance Engineering. EPEW 2014. Lecture Notes in Computer Science, vol 8721 (pp. Page numbers). Springer. https://doi.org/10.1007/978-3-319-10885-8_2

Hron, M., & Obwegeser, N. (2018, January 3). Scrum in Practice: An Overview of Scrum Adaptations.

ISO/IEC. (2011). ISO/IEC WD 25023: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality.

Li, Y., & Manoharan, S. (2013). A performance comparison of SQL and NoSQL databases. In Proceedings title (pp. 15-19).

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.

Nawrocki, P., Wrona, K., Marczak, M., & Sniezynski, B. (2021). A Comparison of Native and Cross-Platform Frameworks for Mobile Applications. Computer, 54(3), 18-27. doi:10.1109/MC.2020.2983893.

Nayebi, F., Desharnais, J., & Abran, A. (2012). The state of the art of mobile application usability evaluation. 2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). https://doi.org/10.1109/ccece.2012.6334930

Nunkesser, R. (2021). Choosing a global architecture for mobile applications. https://doi.org/10.36227/techrxiv.14212571

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]

Rahim, S., Chowdhury, A., Nandi, D., & Rahman, M. (2018). ScrumFall: A hybrid software process model. International Journal of Information Technology and Computer Science (IJITCS), 10(12), 41-48.

Prajapati, M., Phadake, D., & Poddar, A. (2016). Study on Xamarin Cross-Platform Framework. International Journal of Technical Research and Applications, 4(4), 13-18. Retrieved from www.ijtra.com.

React Native. (n.d.). Build Native Mobile Apps Using JavaScript and React. https://reactnative.dev/

Schwaber, K. (1997). SCRUM Development Process. In Business Object Design and Implementation (pp. 117-134).

S. P. Mohanty, D. P. Hughes, and M. Salathé, “Using deep learning for image-based plant disease detection,” Frontiers in Plant Science, vol. 7, p. 1419, 2016.

Soh, J., Copeland, M., Puca, A., & Harris, M. (2020). Overview of Azure Infrastructure as a Service (IaaS) Services. In Microsoft Azure. Apress.

Statista. (Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2022).by region. Statista. https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/

Stack Overflow. (2020). Developer Survey Results 2020. https://insights.stackoverflow.com/survey/2020?#technology-most-loved-dreaded-and-wanted-other-frameworks-libraries-and-tools-loved3

Nielsen,M. (1990).Updating an older interface. Proc. ACM CHI'90 Con! (Seattle, WA, 1-5 April), 243-247

Theocharis, G., Kuhrmann, M., Münch, J., & Diebold, P. (2015). Is water-scrum-fall reality? On the use of agile and traditional development practices. In Proceedings of the International Conference on Product-Focused Software Process Improvement (pp. 149-166). Springer.

VersionOne. (2017). 11th State of Agile Report.

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.