4. System Design

1. **Identification and Diagnosis Pipeline**
    a. Computer Vision Model such as **YOLOv11** to get bounding boxes for leaves and fruits of lemon trees. The bound image will be sent down a pipeline to the next part. As part of the training for leaves and fruits, we could also include identification of various common pests and harmful bugs.
    b. CNN model such as **EfficientNetv2** or **MobileNet** [24] to train on datasets and categorize leaves and fruits to symptoms, dry leaves, dehydrating leaves, deficient leaves.
    c. Large language model such as OpenAI or llama to be trained on lemon tree diagnosis and provide the user with simple actionable plans to treat their orchard. The LLM should **keep in context previous analysis** to account for users who are already properly watering and fertilizing to find undetectable diseases.

2. **Python backend server based on FastAPI**
    a. On demand run the Identification and Diagnosis Model to get recommendation and actionable plants for orchards.
    b. Real time monitoring running the model without the user input with IoT cameras according to pre configured intervals.
    c. Connect to IoT cameras from **Home Assistant** and mobile device cameras for picture inputs.
    d. View and remember past diagnosis and recommendations in LLM context.
    e. Exposed API to allow for connection to other smart agriculture dashboards
    f. Account management using **Google Auth.**

3. **Frontend User Interface based on React**
    a. User Centered Design based on simplicity.
    b. Accessibility to users who are not used to computers and mobile apps.
    c. Access to historical diagnosis and results from real time monitoring.
    d. Diagnosis functions using both smartphone cameras and IoT cameras on demand, and on regular intervals.
    e. Viewing past records function and model analysis on trends to get cues on diseases that aren't visually identifiable. (For example, consistent dehydration despite proper watering)
    f. Configure monitoring settings such as intervals and LLM context, and interface settings such as language.

4. **Data Storage**
    a. Object store tools like (Firebase object store , amazon S3 or Minio) - store images of lemons.
    b. Relational DB - store user accounts and settings, orchard settings, previous diagnosis logs.

4.1 Requirements

5.0 FR Requirements

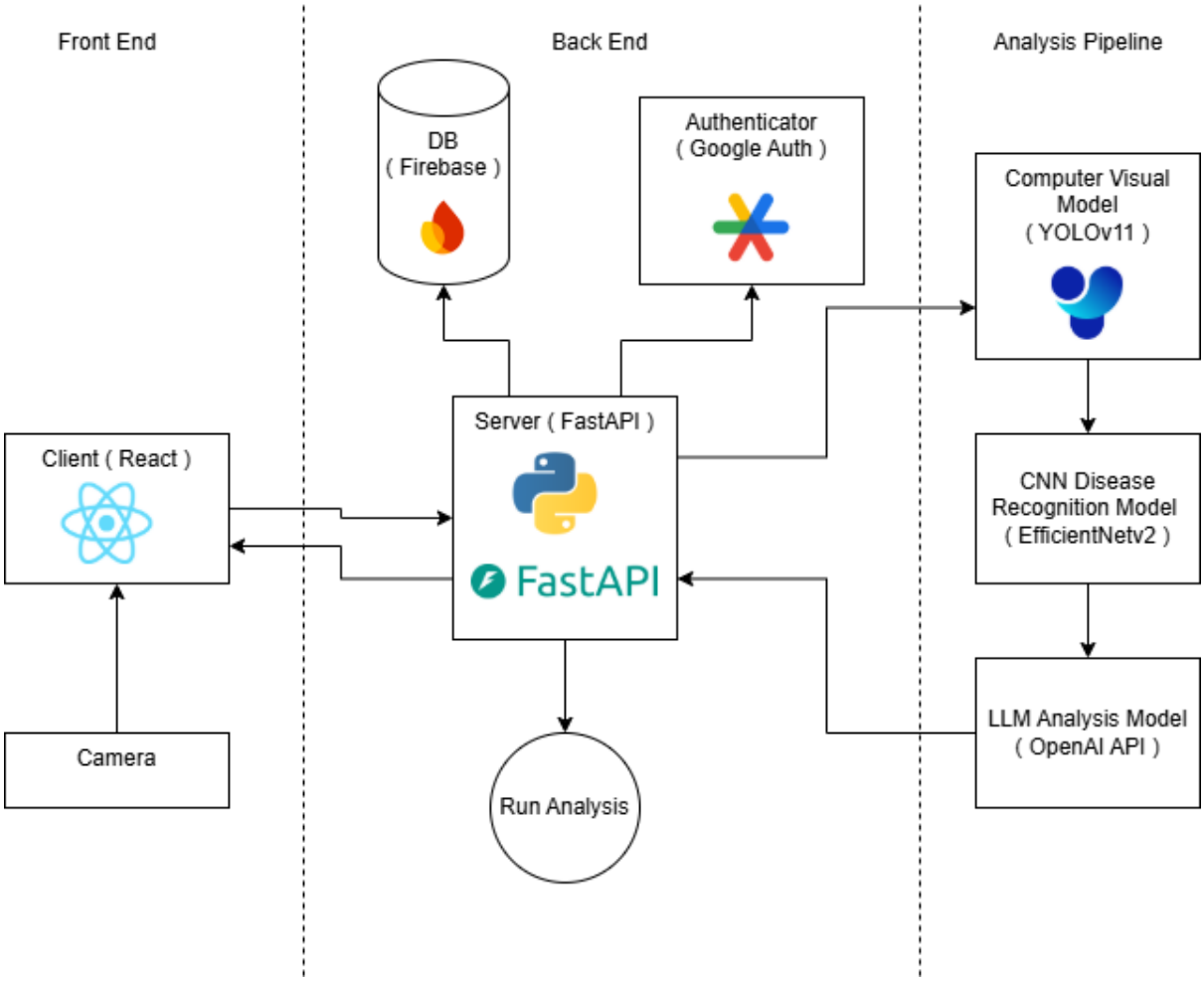
No.	FR Requirement	Description
1	Connect and get image data from smartphone cameras	*The system will allow the user to send smartphone pictures to the identification pipeline*
2	Connect and get image data from IoT connected cameras	*The system will connect and get image data from external IoT cameras using Home Assistant*
3	Upload image for analysis	*The system will allow the user to upload images from their device for analysis*
4	Run identification pipeline on image data	*The system will run and get results from the identification pipeline on image data collected*
5	Provide identification pipeline results	*The system will show users the results of the identification pipeline*
6	Run interval monitoring	*The system will capture image data and run models without user input on pre-determined intervals*
7	Save past results	*The system will save past results from the pipeline to be available for future analysis and viewing*
8	Create and connect accounts	*The system will allow users to create and connect their accounts to the system*
9	Exposed API	
10	Show and change user settings	*The system will allow to show and change user settings including language*
11	Show past results	*The system will allow to fetch data from past results of the user and display them*

The system will allow all its function to be run through an external API for connection to external dashboards

5.0 NFR Requirements

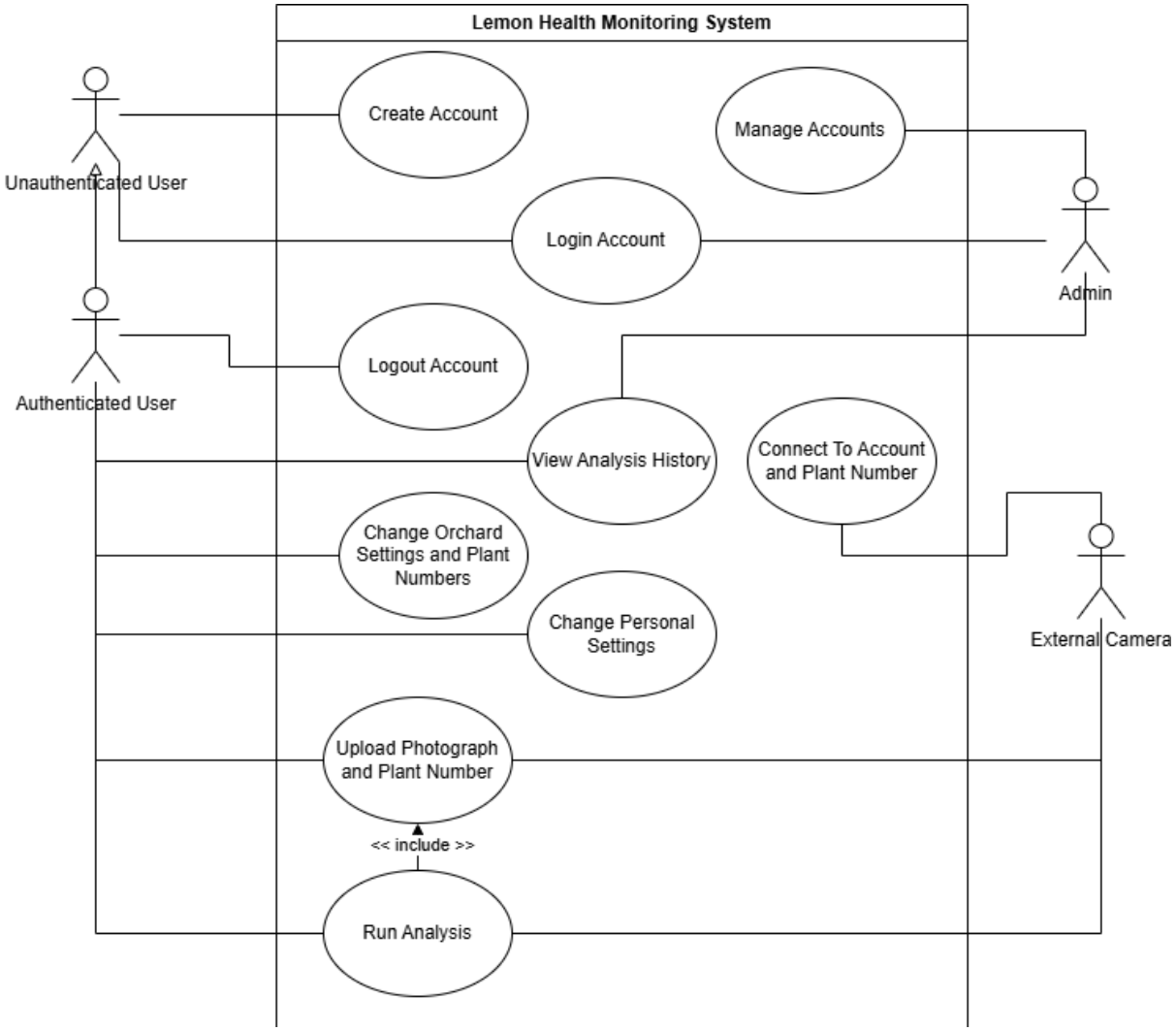
No.	NFR Requirement	Description
1	Model Performance	*The pipeline including all models should finish running within 5 seconds of the user submitting new image data*
2	Model Accuracy	*The pipeline and models shall provide accuracy scores of at least 90% in identifying diseases and symptoms.*
3	Model Scalability	*The pipeline models shall be scalable to more plants and diseases in the future.*
4	Interface Performance	*The user web interface should load and display within 5 seconds on modern smartphones and computers.*
5	Interface Usability	*The user web interface should be accessible even to none computer literate users, SUS score of at least 8*
6	Interface Configurability	*The user shall be able to decide on flexible monitoring intervals and run on demand analysis for image data.*
7	Interface Responsiveness	*The interface shall run on both common tall smartphone displays and common wide computer displays.*
8	Device Compatibility	*The interface shall be accessible and compatible with any device capable of running a full chrome/firefox/edge web browser*
9	Code Documentation	*The system code will be fully documented for ease of maintaining and use*
10	Code modularity and maintainability	*The system code will be modular and maintainable to add additional models and plants to the identification pipeline and the user interface*

4.2 Architecture Diagram



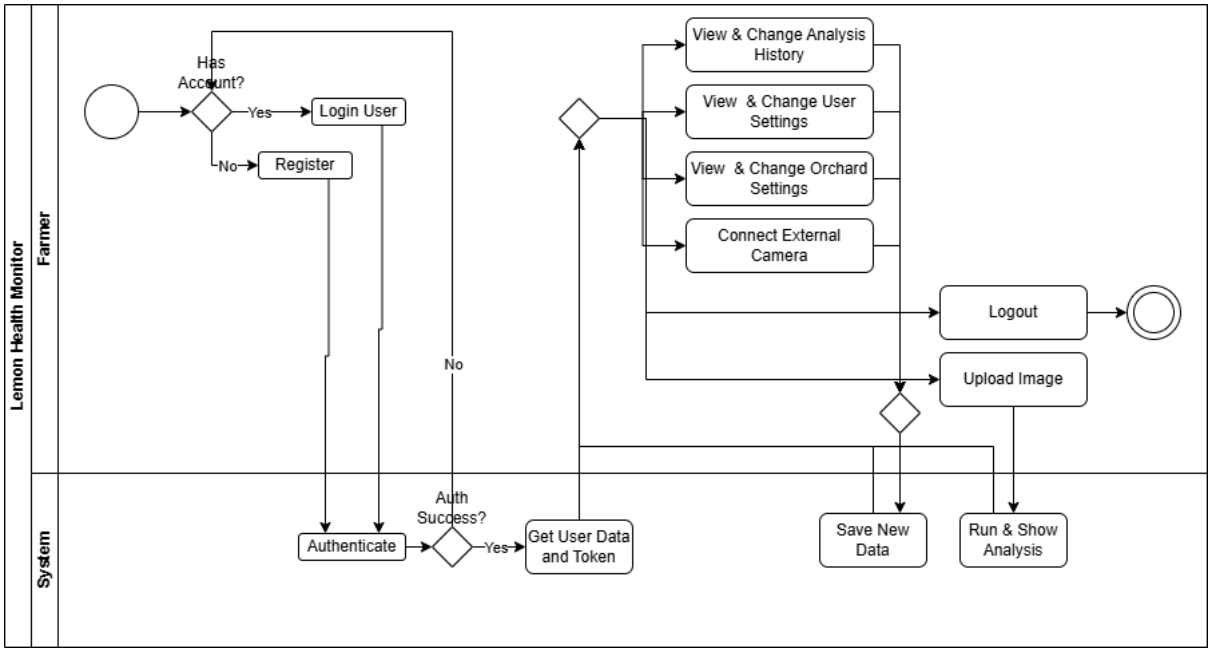
Frontend - Backend architecture design with the analysis pipeline highlighted as a separate section to show a brief of the technologies used.

4.3 User Diagram



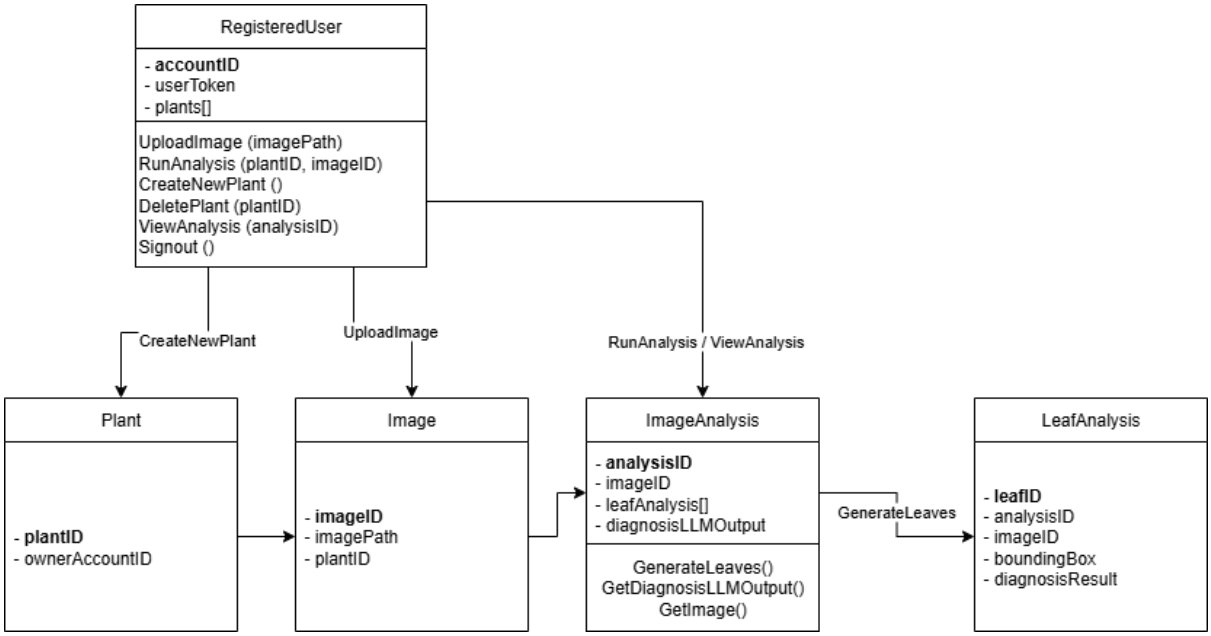
Use case diagram including from top to bottom: account management, personal and orchard settings, uploading of photographs and running analysis, and connecting external cameras through API.

4.4 Activity Diagram



Activity diagram showing user authentication, basic user operations and running an analysis on an uploaded image through the algorithm.

4.5 Objects Diagram



The most important entity of the system that facilitates the core functionality is the ImageAnalysis that will contain the image analysed and a list of all the leaves found by the YOLOv11 model and their individual leaf disease detection model result.

The user will be able to create and remove specific plants by number and upload images to the plant directly through the web interface, or upload automatically through the API.

5. Expected Achievements

a) **Accurate plant analysis -** As a main goal for the project we expect to provide accurate (>90%) analysis of the lemon plant, whether it is simply dehydrated or deficient or whether it has underlying diseases to take care of.

b) **Performative plant analysis -** We aim to deliver results in **under 5 seconds** per image. This will show how efficient our pipeline is and provide the user immediate feedback on what they should do next.

c) **Modular and scalable system -** Both in the sense that the system should be able to support **100 + concurrent users** without performance loss. enable **easy expansion** to include **other crops** (at least 2 more corps).

d) **User centric system -** We aim to make our system as simple as possible for the end user to allow even users without previous technological background to simply scan their plants and get feedback on its status and recommendation on what they should do next. We aim to achieve **SUS usability score ≥ 80 and mobile and desktop accessibility.**

5.1 Challenges

a) **Getting high quality images -** In order to get accurate results it is required the user gets high quality photographs of their lemon trees, as such it would be a challenge to guide the user on getting quality images that would provide the most accurate results.

b) **Training models to get results even on lower quality images -** As a secondary challenge, we would like to make our models work on even lower quality images

5.2 Evaluation

Due to the inherent focus on **artificial intelligence** and machine learning in this project, on the algorithm back-end, we can take standard machine learning metrics to test the viability of our system, this includes **accuracy**, **precision**, **recall** and **F1 scores**.

On the user front-end, we will evaluate the user system usability using a **SUS questionnaire** to assess how accessible the system is to our customer base, mainly farmers and agricultural workers.

A results based oriented metric such as crop yield could potentially be used to assess our system but would require long term usage and data collection on yearly crop yield, before and after implementing our system, and would be susceptible to external effects that may increase productivity or decrease regardless of our system's implementation.

6. Testing

6.1 Testing Approach

Our testing approach will combine both technical validation of the machine learning models and practical validation of the complete system.

6.1.1 Model Testing

Each model in the pipeline (YOLOv11 for detection, EfficientNetV2 for classification) will be evaluated on labeled datasets using standard performance metrics (**accuracy**, **precision**, **recall**, **F1-score**). Stress tests will be performed with varying image qualities (e.g., low resolution, poor lighting) to assess robustness.

6.1.2 Backend API Testing

Unit and integration tests will verify that the FastAPI backend correctly handles image input, executes the analysis pipeline, and returns the correct results. The automated tests will ensure the API remains stable as features expand.

6.1.3 Frontend Testing

Usability and responsiveness of the web interface will be tested across devices. User feedback will provide us with both textual assessments of how using the interface felt to the user and a quantitative metric using SUS to grade our frontend effectiveness as a user-centric design.

6.2 Test Cases

Test	Precondition	Expected Result	Description
#1 UserAuthenticationNewSignup	User has not previous account	Account is created successfully and logged in	1. The user will attempt to authenticate with the Google Auth system 2. The system receives the user token from Google Auth 3. The system successfully creates a new user 4. The system successfully logs the user in and redirects him
#2 UserAuthenticationLogin	User has an account	Account is logged in successfully	1. The user will attempt to authenticate with the Google Auth system 2. The system will find the existing user account and log in, redirecting him
#3 CreateNewPlant	User has an account and is logged into his account	New plant number is successfully created within the system	1. The user will create a new plant number according to how they have arranged their orchard 2. The system will successfully create the plant entity and inform the user of success
#4 DeletePlant	User has an account and is logged into his account, the account has a plant number created	The specified plant is deleted from the system	1. The user will choose which plant to remove 2. The system will successfully remove the plant

#5 Upload Image	User has an account and is logged into his account, the account has a plant number created	The image is successfully uploaded to the system	1. The user will choose the plant number they are taking an image for 2. The user will upload the image from their device 3. The system will successfully save the image and pass on to the analysis pipeline as specified in the architecture
#6 LeafDetection	The leaf detection model is able to run predictions. There is an image of a lemon tree to run prediction on	The leaf detection model has run prediction on an image and returned an accurate list of bounding boxes containing leaves in the image	1. The image will be ran through the leaf detection model 2. The model will return accurate bounding boxes for leaves in the image
#7 DiseaseDetection	The disease detection model is able to run predictions. The leaf detection model has returned bounding boxes for an image	The disease detection model will run on all bounding boxes of leaves and will return accurate results for their diagnosis	1. The image and bounding boxes will be sent to the disease detection run function 2. The model will return an accurate list of results for each bounding box
#8 RecommendationGeneration	The recommendations LLM is able to generate. The disease detection model has run an analysis and returned accurate results.	The recommendations LLM will return in readable text the following: 1. What is the diagnosis of each leaf? 2. How can we verify the diagnosis and what further information is needed to make an accurate diagnosis 3. How to further treat the lemon plant	1. The recommendation LLM will receive the diagnosis of each bounding box from the disease detection 2. The LLM will return accurate and readable text as specified in the test result

#9 AnalysisSequencesFullRun	The user has an account and is logged into his account, the account has a plant number created. The user has an image ready to upload for analysis	The system will run the steps tested in #5, #6, #7, #8 sequentially. The recommendations LLM will return in readable text similar to test #8	1. The user will choose the correct plant number and upload an image for analysis 2. The system will run the image through the leaf detection model to output bounding boxes for leaves 3. The system will run the bounding boxes through the disease detection model to output diagnosis results 4. The system will run the diagnosis results through the generative LLM to provide the user information and recommendations according to specifications in test #8
#10 ViewHistory	The user has ran previous analysis on a plant	The system will show the user the full history of the analysis he has performed on a plant	1. The user will choose a plant 2. The system will fetch and display all previous analysis