Software Engineering Department

Capstone Project Phase B

23-2-D-20

AgarVision: ML-powered app to detect, analyze and count agar plate bacteria colonies.

Authors

Ibraheem Ganayim (Soft. Eng. Department) ✉

Simaan Saada (Soft. Eng. Department) ✉

Supervisor

Dr. Reuven Cohen (Soft. Eng. Department)

Co-Supervisor

Dr. Lilach Iasur Kruh (Biotechnol. Eng. Department)

BRAUDE
College of Engineering, karmiel

Table of Contents

Abstract

We developed a mobile application that automates the process of counting bacteria colonies on given photos of agar plates. Using the application, researchers can take an image of the agar plate and customize the parameters for a tailored analysis. The application provides fast and reliable data, significantly reducing the manual effort needed for colony counting. Testing on the most widely used agar plate in the customer's biotechnology lab resulted in an impressive 98% counting accuracy.

The analysis process begins with adaptive thresholding, which dynamically adjusts image contrast to enhance colony visibility. This step is followed by a custom contour detection analysis that accurately traces each colony's boundaries. Next, the process uses the K-means clustering algorithm to further refine the analysis to distinguish overlapping colonies effectively and to identify non-colony artifacts. This process is required to ensure colony counting reliability.
After pre-processing, the colonies are automatically counted based on the accurately segmented contours, and the results are displayed on the screen.

The app features a scalable server-client architecture with robust security measures. Data is securely stored in the cloud using Firebase database, protected by encryption and by user token authentication. The system has regular data backups to ensure data integrity. By integrating sophisticated image processing techniques, our application streamlines critical research processes, enhancing the efficiency and effectiveness of biotechnological research.

Keywords: Mobile Application, Image Processing, Machine Learning, Bacterial Colony Analysis, Biotechnology Research.

1. Introduction

In the dynamic field of biotechnology research and development, accurate and efficient analysis of experimental results is of utmost importance. Microbiology experiments often involve the use of agar plates, which provide a solid surface for the growth of microorganisms. The traditional method for analyzing these experiments involves manually counting the bacteria colonies located on the agar plates. However, this approach is prone to human error and can be time-consuming, impeding research progress in biotechnology laboratories.

Fig 1-1. Example of an Agar plate.

Despite recent advancements in artificial intelligence, challenges remain in effectively implementing machine learning algorithms for the detection and analysis of bacterial colonies. Therefore, there is a compelling need to develop a solution that overcomes these limitations and provides researchers with a more accurate and efficient means of analyzing agar plates.

Our solution is a mobile app that utilizes advanced image processing algorithms to detect and count bacterial colonies. By leveraging the capabilities of image processing using advanced OpenCV methods such as Hough Transform for Circle and Line detection and machine learning, specifically K-means clustering, we aim to automate the analysis process of agar plates. The implications of our solution are far-reaching, offering significant benefits to researchers in terms of time savings, cost-effectiveness and enhanced accuracy.

The app (AgarVision) has a convenient environment for bacterial experiments and includes the capabilities to count bacterial colonies, save results, preview experiments, add new Agar plates pictures and store data.

Here are some of the app's features:

- **Experiment Management:** Create new experiments and add Agar plates to them.
- **Capture:** The app allows the user to take/upload pictures of samples (Agar Plates).
- **Set Parameters:** Input custom parameters (‘Example’ of a plate and ‘Dilution’ are numbers relevant for the internal use of laborants) for each plate quarter.
- **Count:** The app analyzes the plate image and counts the number of bacteria colonies in it.
- **Save:** The app stores all results safely and securely in the cloud.
- **Review experiments:** The app allows the review of previous experiments' results.

Fig 1-2. App Features
Images for illustration

1. The captured image undergoes adaptive thresholding to enhance colony visibility, followed by contour detection to map colony boundaries.

2. K-means clustering is applied to cluster the colonies into different drops based on their center.

3. The processed data is then automatically counted and displayed, with the option for users to adjust parameters if needed.

2. Background and Related Work

The primary purpose of our customer (Wonder Veggies Ltd. [1]) is to make probiotics plants, especially lettuce, sprouted legumes, and sprouts. In order to evaluate the probiotics in the plants, they need to do experiments and count the number of colonies in each sample.

Wonder Veggies Ltd. [1] perform agar plate tests on a daily basis. In their experiments, they have predefined colors of each bacteria and always-a-color colonies. The process of counting colonies does not need a microscope or UV-integrated devices due to the fact that they can see the colonies after some days of incubation with the naked eye. Usually, they can count them with the naked eye. If not, they call it a “0 colonies count” experiment. Here is a list of techniques available for counting colonies:

2.1. Manually counting

The Agar plate is placed on an illuminated pad and marks each colony with a pen. The mark is registered by a digital application and an audible tone confirms and counts the marks.

Fig 2-1. Counting agar plate manually with a colony counter device.

This process requires humans to detect and mark the colonies t using a pen. This process is time-consuming and sometimes takes up to 10 minutes or more. As

scientists do have a bunch of hundreds of plates, this process will be prone to human error and consume a lot of time.

2.1.1. The counting process issues:

The colony counting method requires counting each dot colony in the plate. Sometimes the petri dish includes more than one experiment. For example, this image shows 4 different tests located in one agar plate, therefore we need to count each experiment individually.

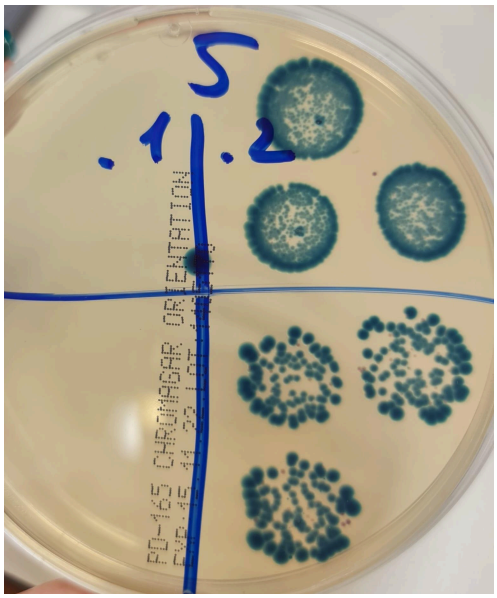


Fig 2-2. Agar plate. Taken in labs.

2.1.2. Sizes of colonies may differ:

We count the number of colonies and their sizes do not change our count. The main difference between different bacteria types is their shape and features. We can see in *Figure 2-4*, two different bacteria, one with a little pink shadow, and another with no shadow. By this feature, we can know that those are different types of bacteria. While in *Figure 2-3*, the colonies have white color, different from *Figure 2-4*'s colonies; this can tell that these colonies are classified as another type of bacteria.

In other words, The size of the colony does not change anything, only the color and its features (shadow, etc…).

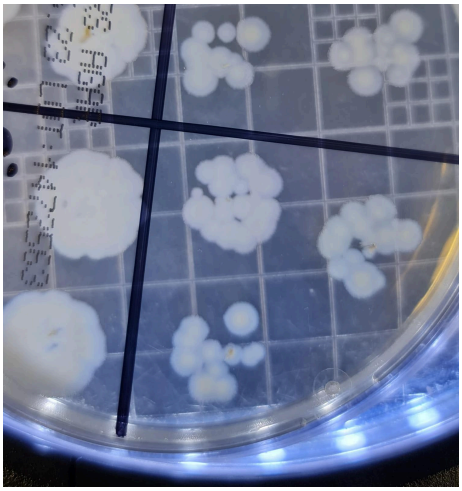


Fig 2-3. Colonies with the same bacteria.

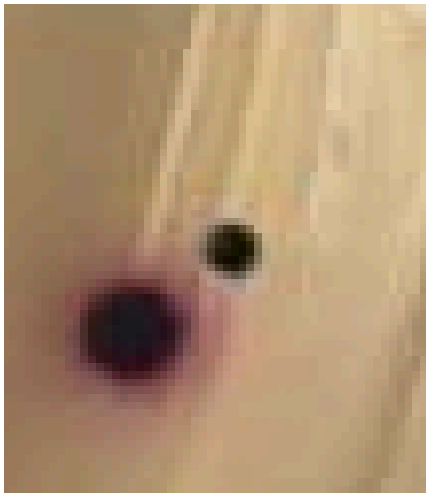


Fig 2-4. Two different bacteria colonies.

2.1.3. **Some colonies cannot be counted:** sometimes, colonies cannot be counted because they are too close to each other (having too much growth) as shown in *Figure 2-5*:

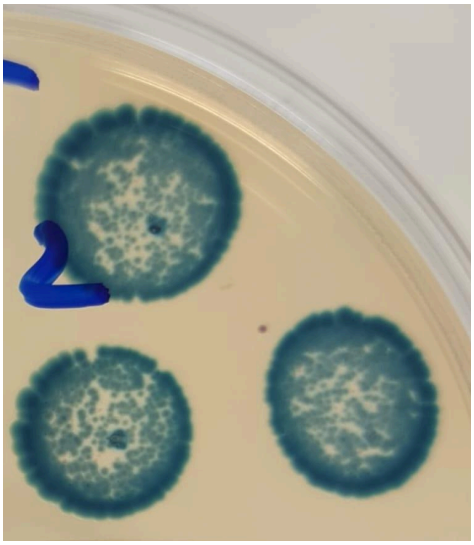


Fig 2-5. These colonies are counted as one "not countable" and not as regular colonies.

2.2. Automated Colony Counter Devices

There are many devices out in the market that can count the colonies with high precision, but they are very expensive and cannot be used by multiple users at once. Here is a list of them:

2.2.1. **Synbiosis ProtoCOL 3 [2]:**

an instrument for colony counting, zone measurements, (inhibition and AST), membranes, and a range of other applications. Cost: ~14,000$.

Fig 2-6. Synbiosis ProtoCOL 3.

2.2.2. **GelCount by Oxford Optronix [9]:**

A dedicated colony counter. All-in-one solution for imaging, counting and characterizing colonies, spheroids and organoids. Cost: ~8,000$.

Fig 2-7. GelCount by Oxford Optronix.

2.3. Software solutions including Image processing, AI & ML

2.3.1. Edge Detection [11]:

Edge detection is a technique of image processing used to identify points in a digital image with discontinuities or sharp changes in the image brightness. The points where the image brightness changes sharply are called the edges (or boundaries) of the internal regions.
As shown in *Figure 2-7*, an image of the coin and its edges in *Figure 2-8.*

Fig 2-7. A coin image.

Fig 2-8. The edges of the coin.

2.3.2. Circle Hough Transform [6,12]:

Circle Hough Transform (CHT), is an essential feature extraction technique for detecting circles in imperfect images. He can work in a 2-dimensional space, with a basic equation where the area of the circle is described by:
$(x - a)^2 + (y - b)^2 = r^2$

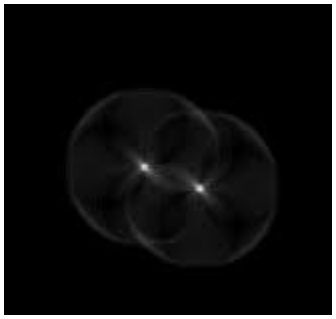


Fig 2-9. Running the CHT algorithm, we can see that in the middle two bold dots are the circle's centers.

This approach is good with its precision. The algorithm is considered to be slow compared with ML and AI algorithms for real-time object detection, like TensorFlow, Yolo(v6 or v7), GoogleML Kit and more.

Many object detection techniques leverage the use of CNN (convolutional neural network), R-CNN (Region-based convolutional neural network) or RNN (Recurrent neural networks) in order to detect objects with high performance. Due to the fact that a convolutional neural network (CNN) or recurrent neural network (RNN) is a type of artificial neural network used primarily for image recognition and processing.

2.3.3. **TensorFlow (object detection):**

Tensorflow has been a leading open-source library for developing ML-based solutions like object detection and image classification.
While we are focusing on object detection. Tensorflow can be a good environment to train and identify the data.

2.3.4. **YOLOv7 [5]:**

You Only Look Once (YOLO)method proposes using an end-to-end neural network application that enables both the predictions of bounding boxes and classification probabilities all at once. YOLOv7 is considered a state-of-the-art real-time object detection algorithm.

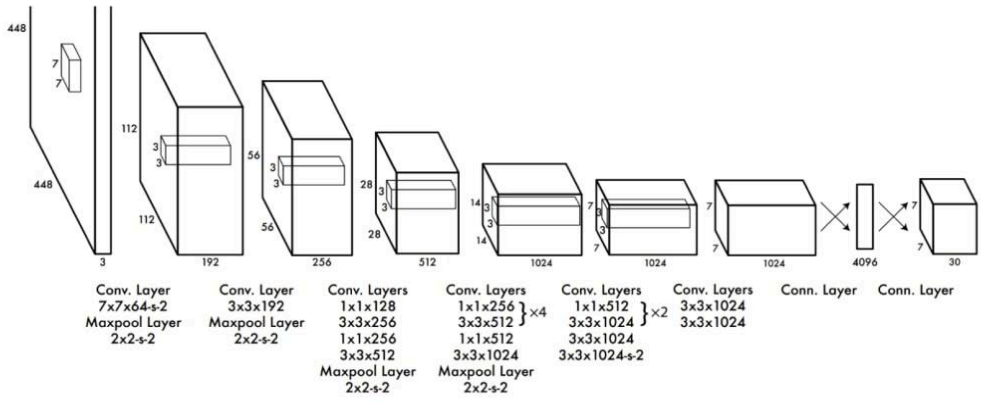


Fig 2-10. YOLOv7 Architecture - Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduces the feature space from preceding layers. We pre-train the convolutional layers on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection.

The strategy followed by YOLO is as follows. First, it divides the given image into an S × S grid. Then, each grid cell is used to analyze whether an object falls into it or not. Hence, each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate this prediction is. The main disadvantage of YOLO is that it is not sensitive to small objects and easy to miss detection.

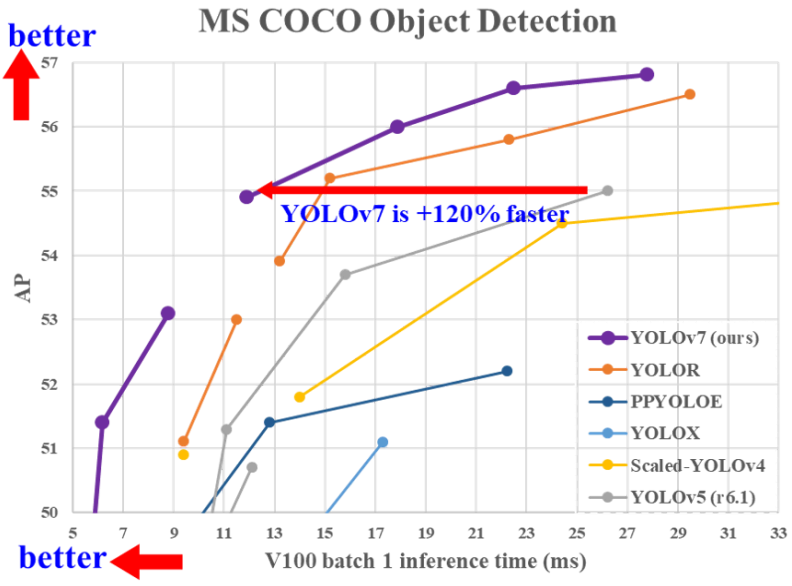


Fig 2-11. Comparison between Different YOLO versions. We can see that YOLOv7 is the fastest.

2.3.5. **Faster R-CNN [14]**:

The algorithm gets the input image and extracts the features of the image using a convolutional neural network (CNN). It uses these features as input to the Region Proposal Network (RPN) to generate potential object-bounding box proposals. Then, scaling the regions into fixed sizes using Region of Interest Pooling (RoI). Then, the fixed-size regions are fed into a fully connected network, which consists of a classifier and a bounding box regressor. The classifier predicts the probability of each proposed region containing an object among a predefined set of classes.

Finally, applying non-maximum suppression to eliminate duplicate detections and refine the final set of objects bounding boxes. This process removes overlapping bounding boxes with lower confidence scores. Keeping the most confident and non-overlapping detections.

2.3.6. **Single-Shot Multibox Detection (SSD) [12]:**

The SSD algorithm was introduced by Wei Liu in 2016. The authors presented a method that could identify objects by using a feed-forward convolutional network by using a single forward pass. Single shot means that in a single forward pass of the neural network (single run of the algorithm), the identification and classification of an object are possible.
The base of the model consists of a VGG-16 convolutional neural network followed by some additional convolutional layers, which reduce the dimensions of the input at each layer.

The network constructs a group that includes all the default bounding boxes within an image and produces the possibility of the presence of an object inside this box by applying convolutional filters to the feature maps that are created. The convolutional layers are applied at different scales in order for the network to be capable of detecting objects of different scales and sizes.
The architecture results in Non-Maximal Suppression (NMS), which is a pruning technique that discards the bounding boxes with a confidence level less than a certain threshold. SSD is known for its precision, compared to YOLO but with the slowest speed [12].

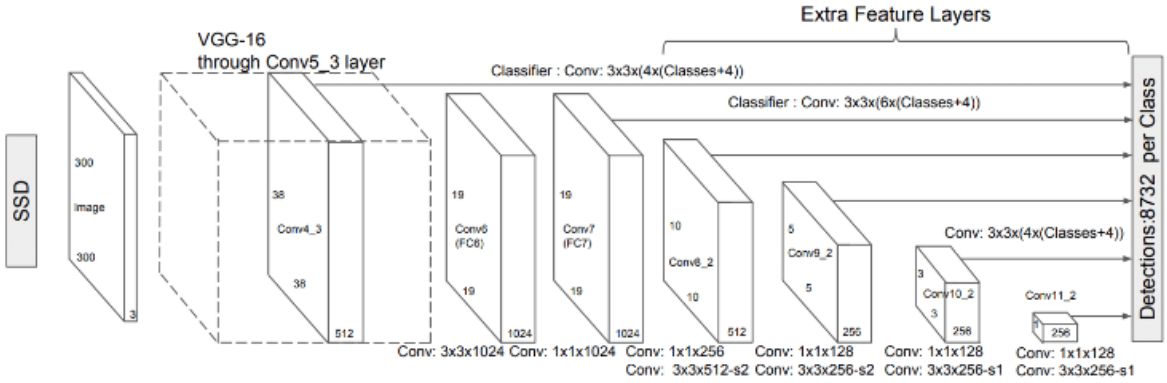


Fig 2-12. SSD Architecture.

3. Solution Architecture

AgarVision's architecture has been designed to optimize performance and scalability, building on the foundations set in Phase A. The system utilizes a server-client architecture, enhanced with robust cloud integration for data management and backup.

3.1. System Architecture

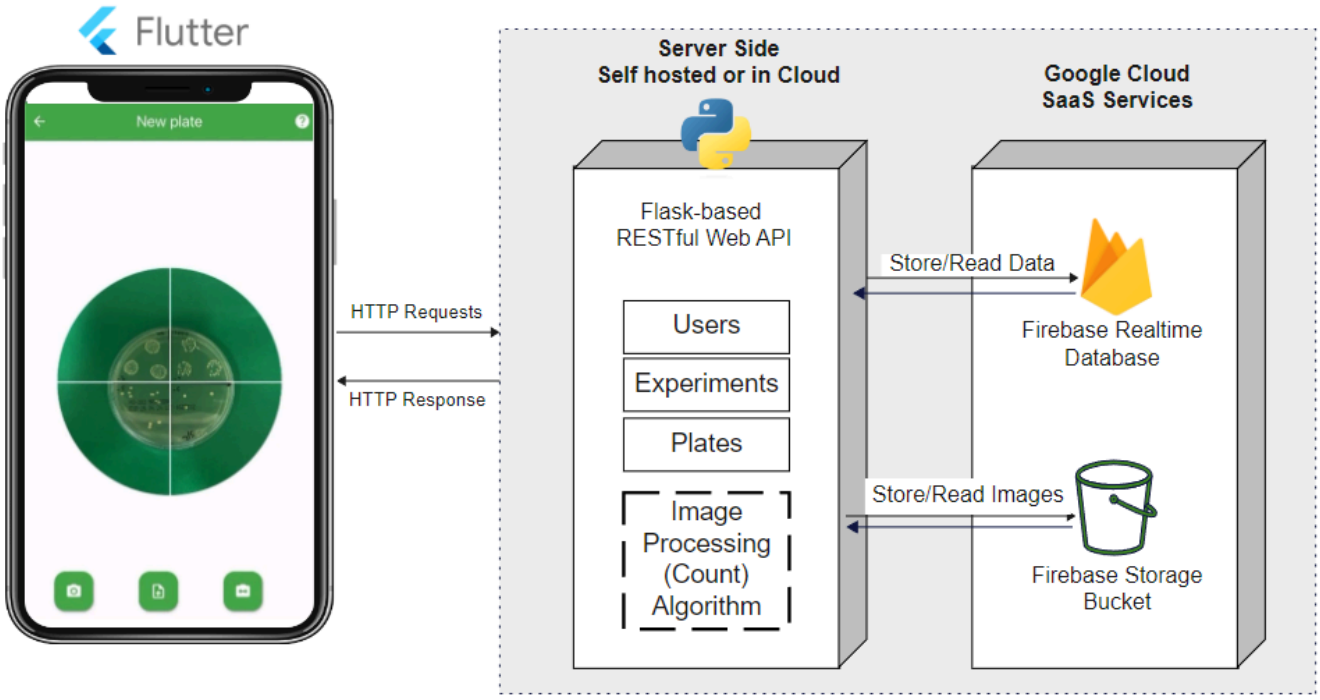


Fig 3-1. System Architecture

The Flask server orchestrates the handling of incoming client requests, which encompasses creating new experiments and new agar plates. It also facilitates the upload and storage of plate images. Data related to experiments is systematically stored in the Firebase Realtime Database, while plate images are securely maintained in Firebase Storage Buckets. Moreover, the server is responsible for robust user authentication processes and efficient management of authentication tokens, ensuring secure and controlled access to system resources.

3.2. Architecture Overview

Software Structure and Operation

The architecture of the software has been meticulously enhanced to integrate advanced image processing capabilities, essential for precise bacterial colony counting.

The Architecture is Comprised of Three Layers:

1. Client-Side Layer:

- The client-side has been developed using the cross-platform framework, facilitating an application compatible with iOS, Android and web platforms. We opted to deploy a Progressive Web Application (PWA) to leverage native performance across all devices, enabling features like camera integration and the handling of large requests to servers.

2. Server-Side Layer:

- The server-side is architected using the Clean Architecture principles, a robust framework recommended by Microsoft for managing dependencies within server-side operations, such as external databases or internal data storage. The server is organized into three primary layers:
    - **WebAPI:** This layer manages all HTTP-based client requests, safeguarding the domain and infrastructure layers from unauthorized access.
    - **Domain:** The central layer houses all domain-specific classes, schemas and code.
    - **Infrastructure:** This layer is responsible for interfacing with external resources, including databases, storage buckets and the prediction algorithm. This structured approach ensures flexibility, allowing easy adaptation from a WebAPI to a CLI application or other communication frameworks with minimal transition time.

3. Database/Storage Layer:

- For enhanced security and scalability, we employ cloud-based database and storage solutions.

This refined software structure not only boosts the application's performance in colony counting but also ensures it remains adaptable and secure, supporting scalable deployment across various platforms and environments.

Core Components:

1. **Image Processing Module:** Now includes adaptive thresholding and custom contour detection algorithms to improve the precision of colony identification and counting.

2. **Data Analysis and Storage:** Utilizes Firebase for real-time data storage and retrieval, allowing for efficient data handling and ensuring data integrity through regular backups.

3. **User Interface (UI):** The UI has been revamped to allow users to easily customize analysis parameters, with real-time display of analysis results and historical data comparison.

3.3. Architecture Features

- **Scalable Infrastructure and Security**

  Our system has been architected to efficiently manage increased data loads, ensuring seamless scalability as user demand escalates. This scalability is facilitated by the modular design principles adopted, allowing for independent scaling of various components.

  On the server side, we have implemented a stateless token mechanism for authentication and authorization. The use of stateless tokens enables the system to scale without the complications associated with login and session storage, as it allows server-side instances to expand in coordination with a Load Balancer. This approach not only enhances the system's ability to handle larger user bases but also maintains robust security and performance without compromising operational efficiency.

- **Enhanced Security**

  Our system architecture has been fortified with advanced security measures, including user token authentication and encrypted data storage, with Firebase serving as the foundational platform for secure cloud storage. Our authentication strategy utilizes two principal types of tokens to identify users who are logged in:

  1. **JWT Access Token**
  2. **JWT Refresh Token**

  Upon successful login, users are issued both tokens; the access token remains valid for 15 minutes, and the refresh token for one month.

  The access token is required for all server-side requests, facilitating entry into the server. Once this token expires, the client side is programmed to request a new access token using the refresh token. This renewal process involves validating the user's credentials again, ensuring they are still authorized to access the system. This method enhances security by reducing the window during which a compromised password can be exploited, as access is contingent on periodic verification of user validity.

  These tokens are securely stored on the client side in secure storage, further safeguarding user data and access credentials.

JWT (JSON Web Token) is a compact, URL-safe means of representing claims to be transferred between two parties. It allows you to encode JSON data that can be verified and trusted with a digital signature. JWTs are typically used for authentication and information exchange, providing a way to transmit data between clients and servers securely.

3.4. Activity Diagram

The following activity diagram illustrates the user interactions with the app and details how data is managed within a meticulously designed, isolated architecture. In this setup, the backend, powered by a Flask server, manages interactions with Firebase and ensures secure data storage. This architecture effectively segregates user interface processes from data handling to enhance security and efficiency.

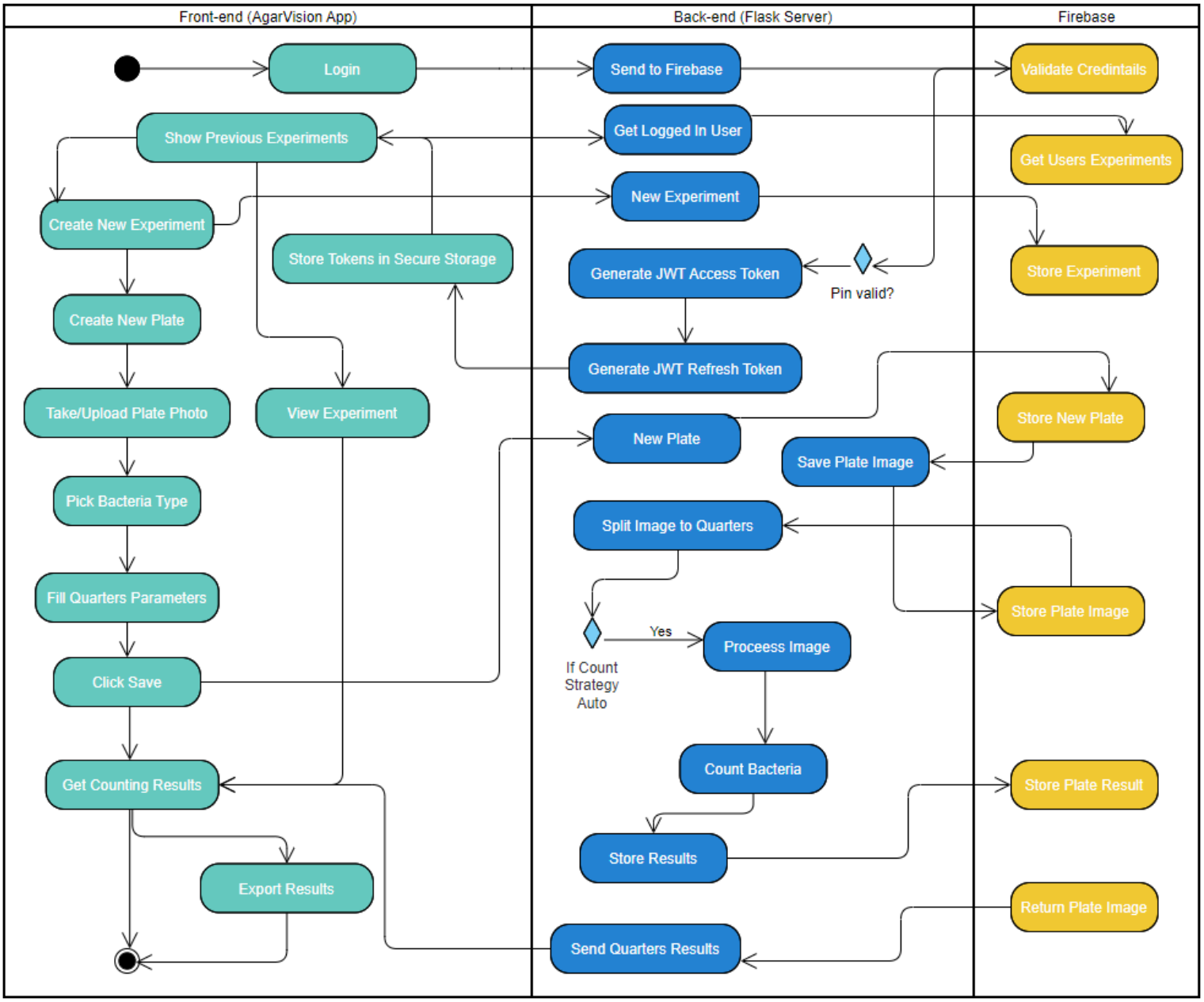


Fig 3-2. Activity Diagram

Here's a breakdown of the activity diagram that maps out the flow of interactions between the front-end (AgarVision App), the back-end (Flask Server) and Firebase in a system designed for counting bacterial colonies on agar plates.

Front-end (AgarVision App)

1. **Login Process:**

- Users log in, and upon successful authentication, their session starts.
- Previous experiments can be viewed immediately after login.

2. **Experiment Management:**

- Users can create new experiments which involve entering details and setting parameters.
- For existing experiments, users can view details and results.

3. **Plate Management:**

- Within an experiment, users can create a new plate by taking or uploading a photo.
- Users select the type of bacteria for the plate.
- Users fill in parameters for each of the four quarters of the plate if the counting strategy is set to manual.

4. **Saving and Processing:**

- After setting up the plate details, users save the data, which triggers the counting process.
- Users can then view the counting results and have the option to export these results.

Back-end (Flask Server)

1. **Authentication and Token Management:**

- The server handles login requests by validating credentials against Firebase.
- Upon successful login, JWT access and refresh tokens are generated and sent to the front-end for secure session management.

2. **Experiment and Plate Data Management:**

- New experiments and new plates are handled by the server, with details stored in Firebase.
- Image data for plates is also managed here, with images being saved and, if necessary, split into quarters for processing.

3. **Image Processing:**

- The plate image can be split into quarters if required.
- The server processes the image to count bacteria, based on the pre-filled parameters.
- Results from the processing are stored and can be sent back to the front-end.

Firebase

- **Data Storage:**
    - Firebase acts as the database where user credentials, experiment details, plate images and results are stored.
    - It is also responsible for the secure handling of JWTs for authentication.

Workflow Interactions

- The workflow typically begins at the front-end with user authentication, followed by either the creation of new experiments or the review of existing ones.

- As users interact with the application by creating new plates and initiating counts, these actions trigger corresponding processes in the back-end where data is processed and stored in Firebase.

- Results of image processing are then relayed back to the front-end where users can review and export outcomes.

This activity diagram efficiently encapsulates the sequence of actions and data flow within the AgarVision application, illustrating the integration between the user interface, server-side logic, and database interactions.

4. Development Process

4.1. Development Methodology

Our development methodology for AgarVision was strategically chosen to facilitate rapid iteration and responsiveness, integrating Agile practices to create an adaptive environment focused on high-quality, quick delivery.

To ensure scientific accuracy, we collaborated closely with domain experts like Dr. Lilach Kroh Isor and her Team, enriching our understanding of bacterial behaviors and microbiological analytics. Identifying technical requirements was crucial, including hardware and software specifications, which guided our selection of tools emphasizing compatibility and scalability. Regular reviews and stakeholder feedback have been pivotal, ensuring our project strategy remains aligned with scientific standards and user needs, continuously refining our approach through an iterative process.

Updated and New Objectives:

- **Algorithm Optimization:** Initially, there was consideration to implement the YOLO machine learning model for colony detection. However, due to insufficient datasets and the infrequent use of the plate types best suited for this model, the focus was adjusted. We redirected our efforts towards refining custom image processing techniques that are better suited to the available data and more commonly used scenarios in our collaborative lab. This optimization was aimed at enhancing the algorithm's ability to accurately identify overlapping colonies and adapt to various lighting conditions and agar plate colors. This strategic pivot ensured that our resources were concentrated on improving technologies that provide the most immediate benefits under the current operational constraints.

- **Enhanced Data Handling:** Using a scalable infrastructure and Implementing advanced data handling capabilities to manage larger datasets more efficiently, allowing users to store, retrieve and analyze historical data seamlessly.

- **Usability and Accessibility Improvements:** Refinements in the user interface to make it more intuitive and accessible for all users, regardless of their technical expertise. This includes simplifying the process of setting parameters for analysis and improving the visual presentation of analysis results.

- **Scalability and Integration:** Enhancing the system's architecture to support scalability and easier integration to ensure that AgarVision can be adopted widely across different research and commercial settings.

Key Aspects of the Agile Methodology Used:

- **Iterative Development:** The project was structured in three-week sprints, allowing the team to focus on delivering specific features or improvements in manageable increments. This approach facilitated regular evaluation and integration of new functionalities.

- **Weekly Stand-ups**: Weekly (and sometimes meetings were held to discuss progress, address any impediments, and realign the team’s efforts with the project goals.

- **Sprint Reviews and Retrospectives:** At the end of each sprint, the team reviewed the work done to assess its alignment with the project requirements and conducted retrospectives to identify areas for improvement in the next sprint.

This methodology not only accelerated the pace of development but also ensured that the team could quickly adapt to changing requirements or address any issues that arose during the development process.

4.2. Research/experiments

Initially, we conducted extensive research and carried out experiments to develop a Minimum Viable Product (MVP) that addresses the problem of counting bacterial colonies on agar plates. Our MVP focused on counting bacterial colonies in an image, even though it has medium-to-high accuracy.

During the research phase, we explored various image-processing techniques and machine-learning algorithms suitable for object detection and counting. We successfully implemented two different types of approaches for our problem: the first approach is based on a hybrid version of CHT [6]. The software we developed allows users to upload a photo of an agar plate and provides a count of the bacterial colonies for each quarter of the plate.

MVP demonstration:

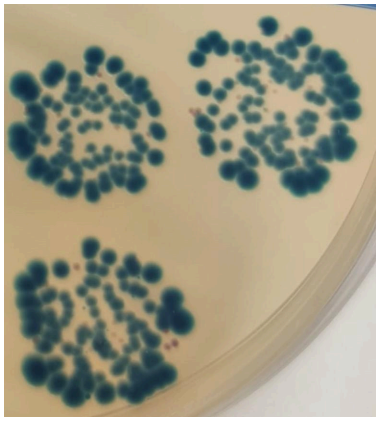
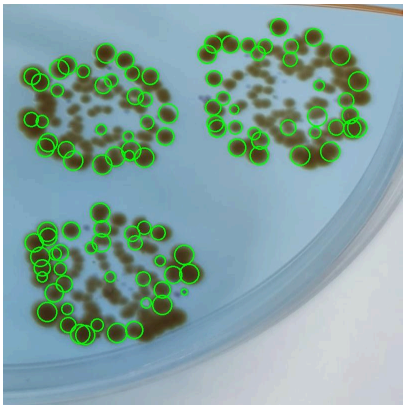
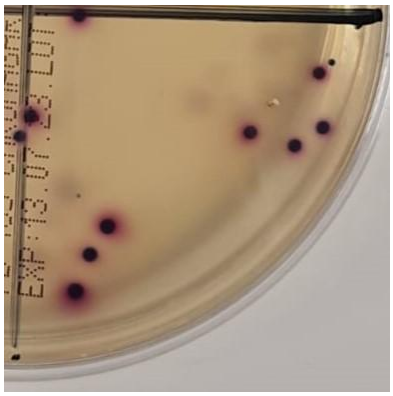
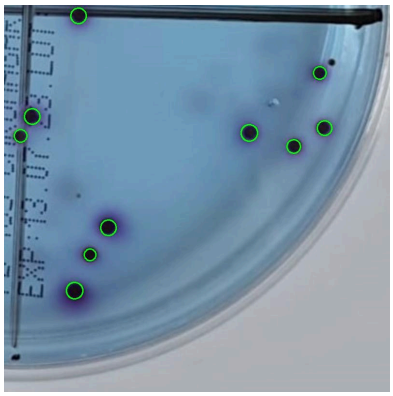
	Input	Output Image	Output Text
1			**110** total colonies
2			**10** total colonies

Fig 4-1 MVP Demo

While the accuracy of our initial MVP was moderate, it served as a promising starting point for further development. Through our experiments, we identified areas for improvement, such as enhancing the image processing algorithm and adding additional preprocessing stages that helped refine the detection algorithm.

These initial research and experiments laid the foundation for our project, demonstrating the feasibility of automating colony counting using image analysis and machine learning. The results of our MVP provided valuable insights into the challenges and complexities involved in accurate colony detection. Building upon this initial success, we had sufficient data to further develop a more advanced and accurate solution for counting bacterial colonies on agar plates.

4.3. Algorithm Components and Workflow

4.3.1. Preprocessing Stage

- **Image Acquisition:** The process starts with capturing/uploading a high-resolution image of an agar plate into the system, serving as the baseline input for subsequent analysis.

- **Adaptive Thresholding:** This technique dynamically adjusts the image's contrast based on localized lighting and color intensity variations, enhancing the visibility of colonies against the agar background.

- **Noise Reduction:** A Gaussian blur filter is applied to smooth the image, reducing noise and removing extraneous details that could interfere with colony detection.

- **Image masking:** Creating masks to highlight bacteria colonies based on their known color. By applying binary masks, only the relevant areas of the image are retained for further enhancing the precision of colony detection and counting.

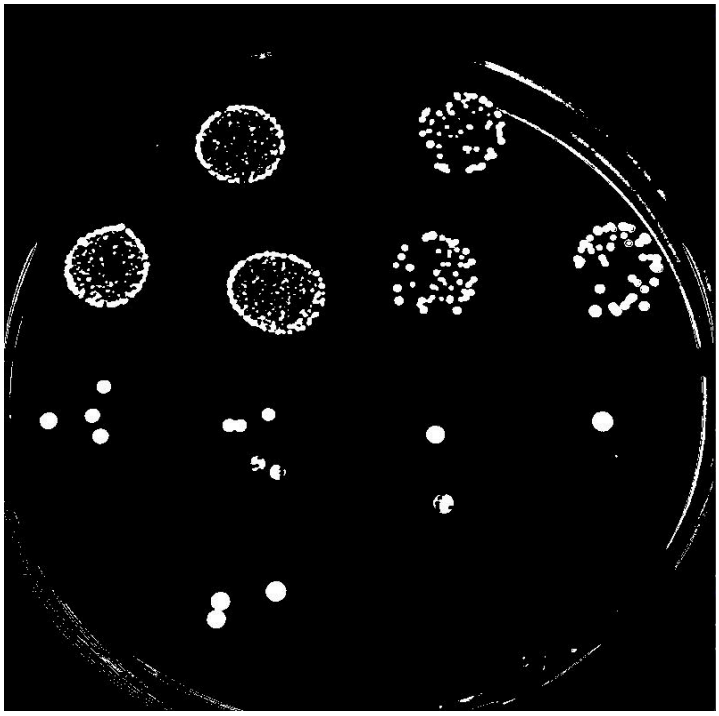


Fig 4-2 - Plate Mask

4.3.2. Quadrant Segmentation Process

- **Image Division:**

  - **Determining Quadrants:** The image is divided into four equal quadrants centered on the image's geometric middle, creating distinct sections for focused analysis.

  - **Purpose of Segmentation:** To run the colony counting algorithm on each segment and determine the count for each one. Since each segment may have different parameters (defined by the user).

- **Individual Quadrant Processing:**

  - **Isolated Analysis:** Each quadrant is independently processed for edge detection, contour detection, and colony identification, reducing computational complexity and enhancing detection accuracy.

Technical Implementation:

- **Edge Detection and Line Drawing:** Techniques like Canny Edge Detection and Hough Line Transform are employed to accurately define quadrant boundaries.

- **Contour Analysis in Each Quadrant:** Independent contour detection within each quadrant ensures efficient management of overlapping regions and minimizes potential errors.

4.3.3. Detection

- **Edge Detection:** Utilizing the Canny edge detector, the algorithm delineates edges within the image, essential for identifying potential colony boundaries.

- **Contour Detection:** OpenCV's findContours method extracts contours from the edged image, with each contour potentially representing a colony.

- **Custom Shape Analysis:** Each contour is further analyzed using cv2.approxPolyDP, simplifying it to fewer vertices and determining its potential as a colony based on shape complexity.

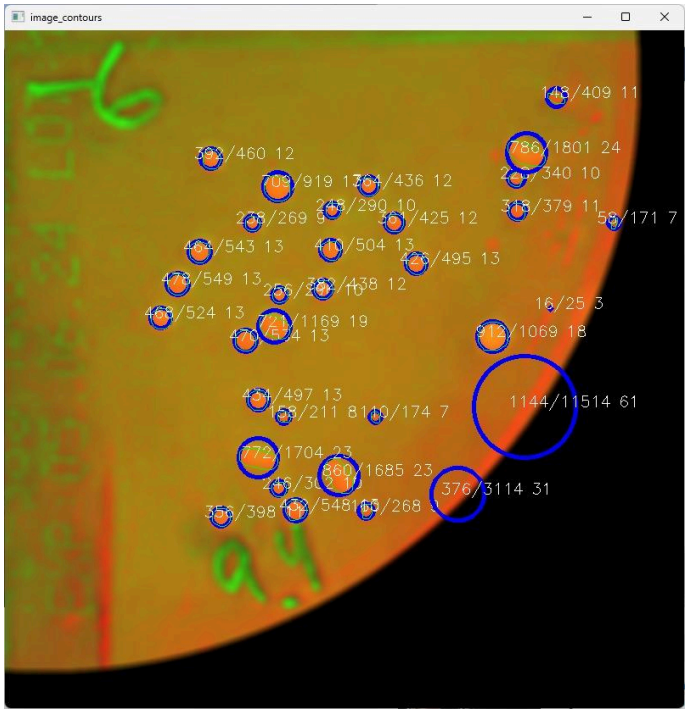


Fig 4-3
The underlying analysis of our algorithm to check whether the current circle is false positive or not, we can see written on the side of each circle the area of the current circle and the area of found colony/shape. For each detected circle we calculate the minimum enclosed circle and then get the area and the radius of the circle, and if the area of the hull convex of the contour of the current shape was bigger than 95% of the minimum enclosed circle, we filter this and can verify that this is a colony with percent 95%.

4.3.4. Colony Segmentation Using K-Means Clustering

As we can see in Figure 6 below, Each quadrant has three main clusters corresponding to the 3 bacteria drops in each quarter; this technique is necessary to validate the accuracy of bacteria growth. The colonies are clustered using the K-means algorithm based on a predefined number of centers, to count each drop separately.

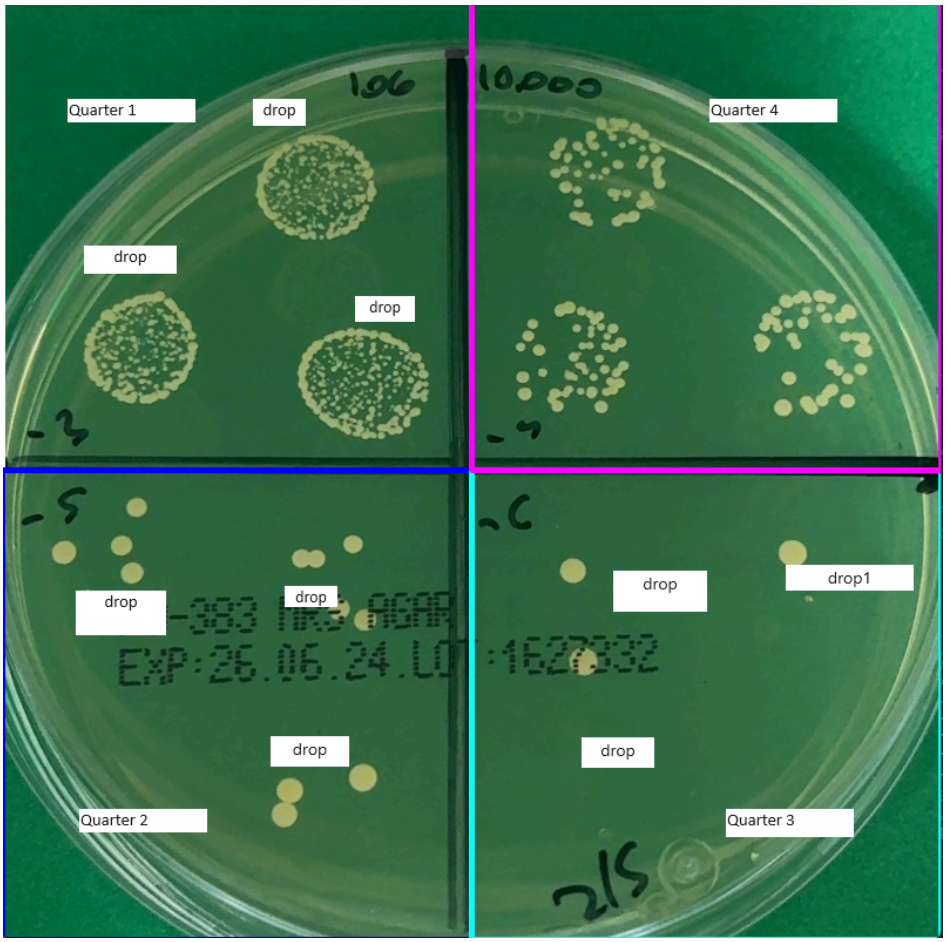


Fig 4-4 - Image Segment Details

4.3.5. Integration with User Interface

- **User Interaction:** Users engage with the system through a Flutter-developed application, facilitating image uploads and result management.

- **Data Handling:** The Flask-developed backend manages interactions with Firebase, ensuring secure and efficient data storage and retrieval.

4. Tools and Technologies Used

The selection of tools and technologies was meticulously curated to support the sophisticated image processing and robust data management needs of the AgarVision project:

1. **OpenCV:** Employed for implementing advanced image processing functions, particularly adaptive thresholding and custom contour detection, essential for the accurate segmentation of bacterial colonies.

2. **Firebase:** Selected database solution (as the primary Cloud service from Google Cloud platform) for its capabilities in handling real-time data efficiently. Firebase's robust security features and seamless integration with various platforms make it ideal for our application's needs, including the Firebase Realtime Database for data storage and Firebase Storage Bucket for managing image files.

3. **Flutter:** Utilized for developing a cross-platform mobile application, ensuring a consistent and seamless user experience across Android, iOS, and web platforms. The application is containerized using Docker, enhancing its portability and scalability across different environments.

4. **Python and NumPy:** These are the primary programming languages and libraries used for backend algorithm development. Python’s versatility and NumPy’s computational efficiency are crucial for implementing the custom K-means clustering algorithm and other image-processing functionalities.

5. **GitHub:** Acts as the version control system to manage codebase changes and foster collaboration among the development team, ensuring that updates and integrations are handled smoothly.

6. **JIRA:** Employed for task tracking, bug tracking, and monitoring sprint progress, which is essential for effective project management under the Agile methodology.

7. **Docker:** Used to containerize both the Flutter-based client application and the Flask server. Docker encapsulates the application environment, ensuring consistency across development, testing, and production deployments.

8. **Heroku:** Deployed as the cloud provider for hosting both the client and server components of our system. Heroku's platform facilitates easy deployment and scaling of containerized applications managed via Docker CLI, providing a robust infrastructure for our server needs.

9. **Heroku & Docker CLI:** These command-line interfaces are used to manage application deployments and orchestrate container behavior, ensuring streamlined updates and maintenance of the system.

10. **Libraries for Algorithm Implementation:** The previously provided algorithm utilized libraries such as '**imutils'** for image manipulation, '**matplotlib'** for plotting, and '**sklearn'** for applying the K-means clustering algorithm. These libraries were critical in enhancing the application's analytical capabilities to deliver precise and reliable results.

These tools and technologies were not only chosen for their robustness and capability but also for their widespread adoption and support within the developer community, guaranteeing that our team had access to extensive resources and support for effectively tackling the project's challenges.

5. Challenges and Solutions

5.1. Technical and Engineering Challenges

We encountered several nuanced challenges, especially related to optimizing the detection strategies for different types of agar plates, namely MRS and Chrome Agar.

5.1.1. Chrome Agar Plate Detection

Challenge: We initially explored using the YOLOv5 model for detecting bacterial colonies on Chromagar plates, aiming for a robust automated system that could handle different agar types. However, two main issues surfaced: the infrequent use of Chrome Agar in our primary collaborative lab and the need for a substantial dataset to effectively train the deep learning model.

Solution: Given these limitations, we decided to pivot our focus away from Chrome Agar and concentrate on optimizing detection for MRS agar, which was more commonly used in our collaborator's operations. This decision allowed us to allocate our resources more efficiently towards enhancing detection techniques where they would be most impactful and used.

5.1.2. MRS Agar Plate Detection

Challenge: MRS agar, being more frequently used, required reliable and precise detection strategies that could be efficiently implemented without the complexities of deep learning models, given our dataset constraints.

Solution: We opted to enhance our existing image processing algorithms for MRS agar detection. By refining adaptive thresholding and custom contour detection techniques and hough transform for circle detections, as outlined in the algorithm provided earlier, we were able to achieve high accuracy without the need for extensive datasets required by more complex models like YOLO.

6. Results and Evaluation

Our goal was to develop and deploy a mobile application capable of automating the counting of bacterial colonies on agar plates with high accuracy and user-friendliness. This goal was set with the aim of significantly reducing the manual effort required in biotechnological research labs and improving the reliability of colony counting processes.

1. Deployment of a Working Mobile Application:

We successfully developed and deployed a mobile application that enables researchers to take photos of agar plates and automatically receive colony counts, achieving a remarkable 98% accuracy in real-world lab environments, particularly on MRS agar plates. This high level of accuracy is a result of meticulous refinement of image processing algorithms, including adaptive thresholding and custom contour detection. The strategic decision to focus primarily on MRS agar, guided by its prevalence in our collaborator's lab, enabled the optimization of the algorithm, for the most commonly used agar type.

2. User-Friendly Interface for Researchers:

The application was designed with a strong focus on user experience, featuring an intuitive interface that simplifies the process of capturing images and viewing results. Initial user feedback has been overwhelmingly positive, with researchers praising the app's simplicity and effectiveness. This success is attributed to the development team's use of Agile methodologies, which enabled rapid iterations of the user interface based on real user feedback. This user-centered approach ensured that the application was not only functional but also closely aligned with the needs and preferences of its end users.

6.1. Accuracy Analysis

In assessing the performance of our AgarVision algorithm, we conducted extensive testing on a set of 60 different MRS agar plates. The results from these tests yielded an average accuracy rate of 95%, with a minimum accuracy threshold of 90%. This indicates a robust performance of our algorithm across varied test conditions.

Standard Deviation Calculation:

To further quantify the variability in our accuracy measurements across the different tests, we calculate the standard deviation of the accuracy rates. The standard deviation provides insights into the consistency of our algorithm. The calculation is performed using the following formula:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}}$$

Where:

- σ represents the standard deviation,
- x_i denotes each individual accuracy measurement,
- μ is the mean accuracy (95% in our case),
- n is the total number of observations (60 agar plates).

$$\sigma \approx 1.41\%$$

The standard deviation helped us understand the spread of accuracy rates around the mean, providing a clearer picture of the algorithm's reliability across multiple samples. A lower standard deviation would indicate that the accuracy rates are more consistently close to the average, underscoring the stability and reliability of our algorithm in practical applications.

Key Areas of Stakeholder Input:

- **Feature Feedback:** Direct feedback from biotechnological researchers who used the application in a beta testing phase was instrumental in fine-tuning the UI and functionality. Their insights led to significant enhancements in the app's analytical features, such as the addition of customizable parameters for analysis.

- **Performance Evaluation:** Stakeholders from the customer's biotechnology lab provided data and reports on the application's performance, particularly its accuracy in different testing scenarios. This helped identify and rectify specific issues with image processing algorithms.

- **Usability Suggestions:** Feedback on the app's usability led to several user interface improvements, making the app more intuitive and easier to navigate for users with varying levels of technical expertise.

The interactions with Wonder Veggies were crucial for refining AgarVision. These were conducted through regular update meetings and feedback sessions, utilizing collaborative tools that enabled stakeholders to directly annotate and provide suggestions on specific application elements. This continuous dialogue ensured that the development of AgarVision stayed well-aligned with user needs and expectations, driving the project toward a successful outcome.

6.2. Tests

Login Page:

#	Test ID	Description	Excepted Result	Comments
1	successfullLogin	Press on “login”.	Redirect to HomePage “View All Experiments” and save credentials to devices persistent storage.	After inputting username and password.
2	failedLogin	Press “login” with wrong or non-existing credentials.	Display error message and stay at the login page.	After inputting a wrong or non-existing username or password.
3	emptyUserName	Press the “Login” button.	Display appropriate messages.	After clicking “Login” when the username field is empty.
4	emptyPassword	Press the “Login” button.	Display appropriate messages.	After clicking “Login” when the password field is empty.

View All Experiments Page:

#	Test ID	Description	Excepted Result	Comments
1	viewExpirements	First page after init the app.	Display all experiments conducted by the logged in user.	First page after successful login too.
2	newExpirementSuccess	Click on the “New experiment” button.	Display a new experiment page.	
3	noCameraPermission	Checked when the “new experiment” page initiated	Display “need permission” prompt.	With two buttons, accept and deny.
4	grantedCameraPermission	When “new experiment” page initiated	Display the real time camera to the user.	take the agar plate picture in this process
5	takenPictureSuccess	When the user takes picture and press “Done”	Display “Select Boundaries page”	The user can select the boundaries of the plate.
6	takenPictureFailed	When user press “Cancel” in camera	Return to page “All Experiments”	

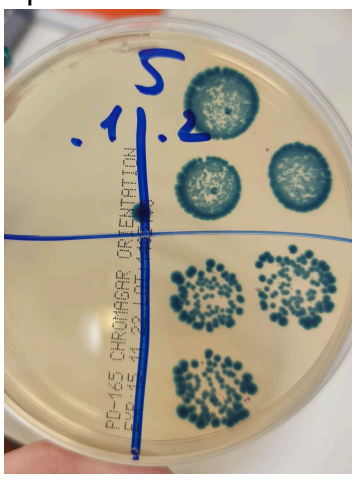
Select Boundaries Page:

#	Test ID	Description	Excepted Result	Comments
1	successDisplayImageIn Viewer	Checked when page initiated	Display the taken image in the viewer.	Default behavior.
2	failedDisplayImageInVie wer	Checked when page initiated	Display appropriate message. And go back to take new picture page.	
3	boundariesNotSelected	Checked when click “finish” button	Display appropriate message.	Stay in same page.
4	boundariesSelectedSuc cess	Checked when click “finish” button	Close this page and continue to “Set Experiment Params Page”	

Set Experiment Parameters Page:

#	Test ID	Description	Excepted Result	Comments
1	emptyBacteriaTypeField	Checked when click “finish” button	Display appropriate message.	Stay in same page.
2	emptyDilutionField	Checked when click “finish” button	Display appropriate message.	Stay in same page.
3	emptyColorField	Checked when click “finish” button	Display appropriate message.	Stay in same page.
4	successSetParams	Checked when click “finish” button	Close this page and go back to the last page.	

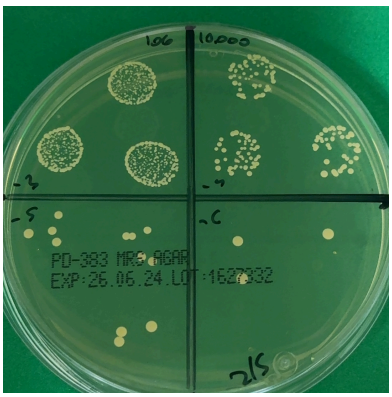
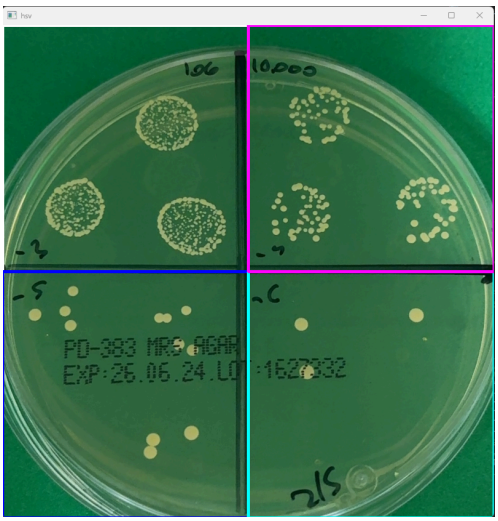
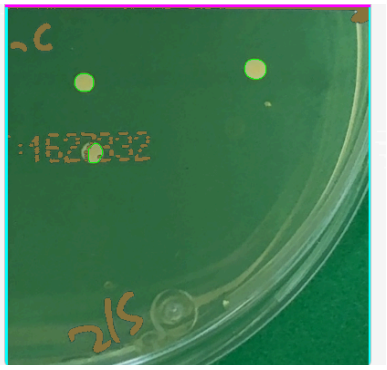
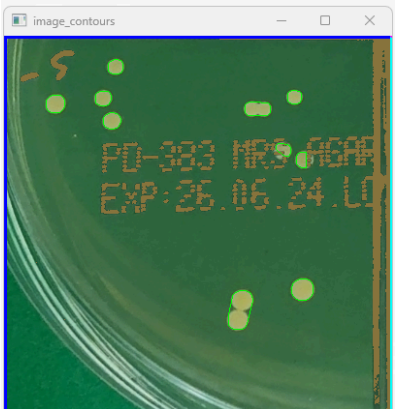
Results Experiment Page:

#	Test ID	Description	Excepted Result	Comments
1	Input: 	Display the plate image in the top viewer	Display the plate image in the top viewer.	Stay in same page.
2	countingProccessSuccess	Result from Server-side	Display the quarters list in the bottom panel of the page.	Stay in same page.

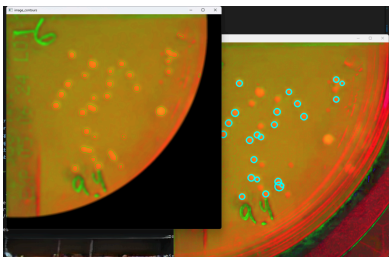
#	Test ID	Description	Excepted Result	Comments
3	waitingForCoutingProccessToFinish	Result from Server-side	Display loading message.	Stay in same page.
4	countingProccessFailed	Result from Server-side	Display error message, save the params and the image to db. With no results.	The user can re-count when entering the “Experiment Page”

Counting Process (Server Side):

#	Test ID	Description	Excepted Result
1	Input: 	Count the quarter.	0 colonies.
2	Input:	Divide the agar plate to 4 quarters.	4 quarters.

#	Test ID	Description	Excepted Result
			
3		Count another quarter	3 colonies detected out of 3.
4		Count another quarter	12 colonies detected out of 12.

BRAUDE
College of Engineering, karmiel

#	Test ID	Description	Excepted Result
5		Count another quarter	27 colonies detected out of 32.

7. Lessons Learned

7.1. Reflections and Improvements

Development of the AgarVision app provided numerous learning opportunities and insights into both the technical and project management aspects of software engineering within a scientific application context. Here are some key reflections and areas for potential improvements:

1. Importance of Targeted Algorithm Optimization:
Focusing our efforts on enhancing image processing algorithms specifically tailored to MRS agar plates proved to be highly effective, allowing us to achieve high accuracy without the extensive resources that deep learning models would require. This strategic focus was pivotal in our success. Looking ahead, exploring hybrid approaches that combine lightweight machine-learning models with traditional image processing could offer significant benefits. Such approaches would be particularly advantageous as we expand to include other types of agar or more complex colony characteristics in future phases of development.

2. Agile Methodology in Scientific Software Development:
Using Agile practices helped us stay flexible and responsive to user feedback, crucial for developing a user-centric application. This approach allowed for rapid iterations, enhancing the app's usability and effectiveness. To further improve our understanding of user needs, we could benefit from increasing the frequency and depth of user testing sessions. This could reveal additional areas for enhancement not immediately obvious to developers or occasional users.

3. Data-Driven Decision Making:
Decisions to focus on specific agar types and optimization strategies were effectively guided by data from usage patterns and feasibility studies, ensuring that resources were wisely invested for maximum impact. Implementing more advanced data analytics within the app could further enhance this approach, providing ongoing insights into usage patterns and accuracy, and allowing for continuous refinement of the application based on actual user data.

8. Algorithm Accuracy Evaluation

The accuracy of the AgarVision algorithm is evaluated by comparing the automated colony counts against manual counts, which serve as the gold standard. The specific steps involved in the accuracy calculation include:

1. **Automated vs. Manual Counting:**

- Both automated and manual counts are performed on the same set of agar plates to ensure consistency in the comparison.

- The counts from the AgarVision algorithm are statistically compared to the manual counts. Accuracy is typically expressed as a percentage using the formula:

$$\text{Accuracy} = \left(\frac{\text{Number of Correct Detections by the Algorithm}}{\text{Total Number of Colonies Counted Manually}} \right) \times 100$$

2. **Validation and Error Analysis:**

- **Error Analysis**

  Discrepancies between automated and manual counts are analyzed to identify common sources of error, such as missed colonies, overcounting, or misidentification.

- **Continuous Refinement**

  Based on the error analysis, the algorithm parameters (e.g., threshold levels, contour approximation accuracy) are continuously refined to improve the accuracy.

AgarVision's algorithm accuracy is a critical metric that confirms the application's reliability for scientific research. Through meticulous calibration and continuous testing against manual counting methods, AgarVision ensures that its automated counts are both accurate and dependable.

9. References

[1] Wonder Veggies Ltd. the company we are building the tool to. http://wonderveggies.co/

[2] ProtoCOL 3 - Automatic colony counting and zone measuring

[3] The Importance of Colony Counting. Colony Counters - Bacteriological Tools | Weber Scientific.

[4] Colony Forming Units — What Are They and How Many Do You Need?. Colony Forming Units: What Are CFUs and How Many Do You Need?

[5] YOLOv7 object detection. GitHub - WongKinYiu/yolov7: Implementation of paper - YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors

[6] Circle Hough Transform. Circle Hough Transform - Wikipedia

[7] Google ML kit. ML Kit | Google for Developers

[8] Han, S., Lee, S., Lee, J., Lee, S., Lee, H., Lee, H., & Park, S. (2021). Automatic colony detection using machine learning and image processing for high-throughput screening. Scientific Reports, 11(1), 1-10. https://doi.org/10.1038/s41598-021-83591-1

[9] GelCount™ Mammalian-cell colony, spheroid and organoid counter

[10] Yuen, H. K., et al. "Comparative study of Hough transform methods for circle finding." Image and vision computing 8.1 (1990): 71-77.
Comparative study of Hough Transform methods for circle finding - ScienceDirect

[11] Shrivakshan, G. T., and Chandramouli Chandrasekar. "A comparison of various edge detection techniques used in image processing." International Journal of Computer Science Issues (IJCSI) 9.5 (2012): 269.
https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=685e8ecc440343d59f157ca1add377b956b2199d

[12] Tan, L., Huangfu, T., Wu, L. et al. Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification. BMC Med Inform Decis Mak 21, 324 (2021). https://doi.org/10.1186/s12911-021-01691-8

[13] D. Kreuzberger, N. Kühl and S. Hirschl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," in IEEE Access, vol. 11, pp. 31866-31879, 2023, doi: 10.1109/ACCESS.2023.3262138.
Machine Learning Operations (MLOps): Overview, Definition, and Architecture | IEEE Journals & Magazine

[14] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems 28 (2015).

10. User Guide - AgarVision App

AgarVision is a flutter-based mobile application designed to automate the detection and counting of bacterial colonies on agar plates. Utilizing advanced image processing and machine learning algorithms, AgarVision offers biotechnology researchers a fast and accurate method to analyze bacterial colonies, enhancing productivity and reducing manual counting errors.

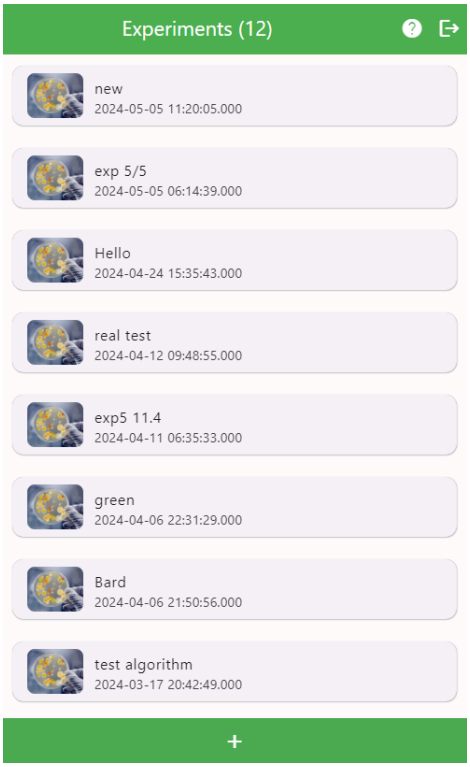
Getting Started

1. **Installation:**
   - Open this URL from any browser https://agarvision-f1132e30aa54.herokuapp.com/
   - Optionally:
     - Add the website to the home screen to enable PWA features.
     - Now the website is added as “AgarVision” as PWA (Progressive Web Application).

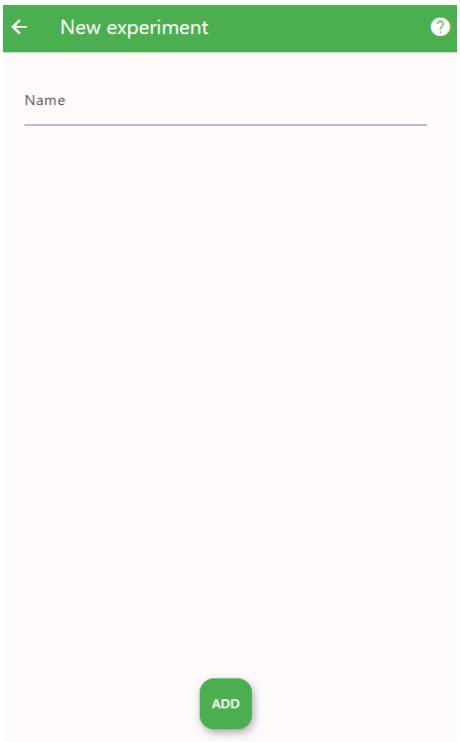
2. **Login:**
   - Open the AgarVision app.
   - Enter your email and password to log in.
     - Demo User:
       - Email: demo@agarvision.com
       - Password: demo1
   - If you are a new user, please contact your company to register a new user.

Navigating the App

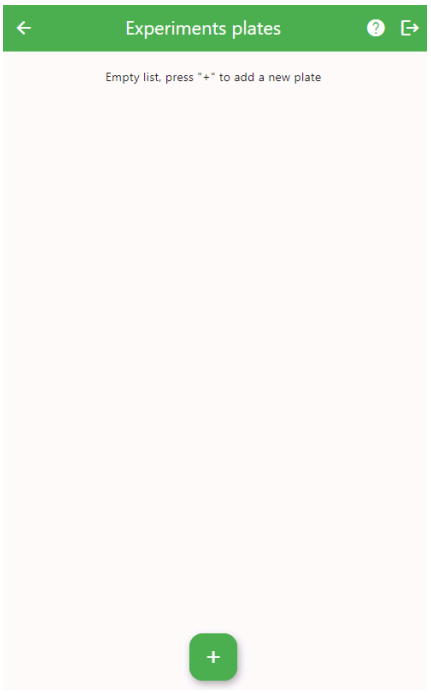
1. **Dashboard:**
    - Upon successful login, you will see a dashboard displaying all previous experiments.
    - You can review past experiments or create a new one.



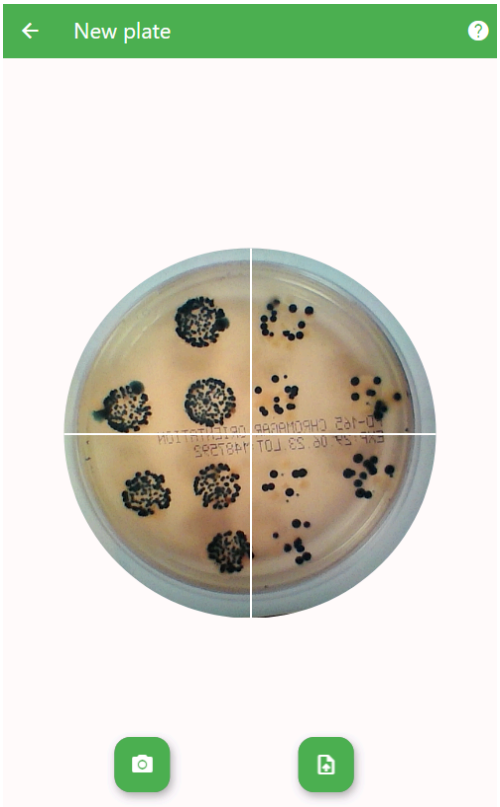
2. **Creating a New Experiment:**
    - Tap the plus (+) button located at the bottom center of the dashboard.
    - Enter a name for your new experiment and the “ADD” to proceed to an "Empty Plates" page.



3. **Adding a New Plate:**
   - In your experiment view, tap the plus (+) button to add a new plate.
   - You will be directed to a camera preview page.

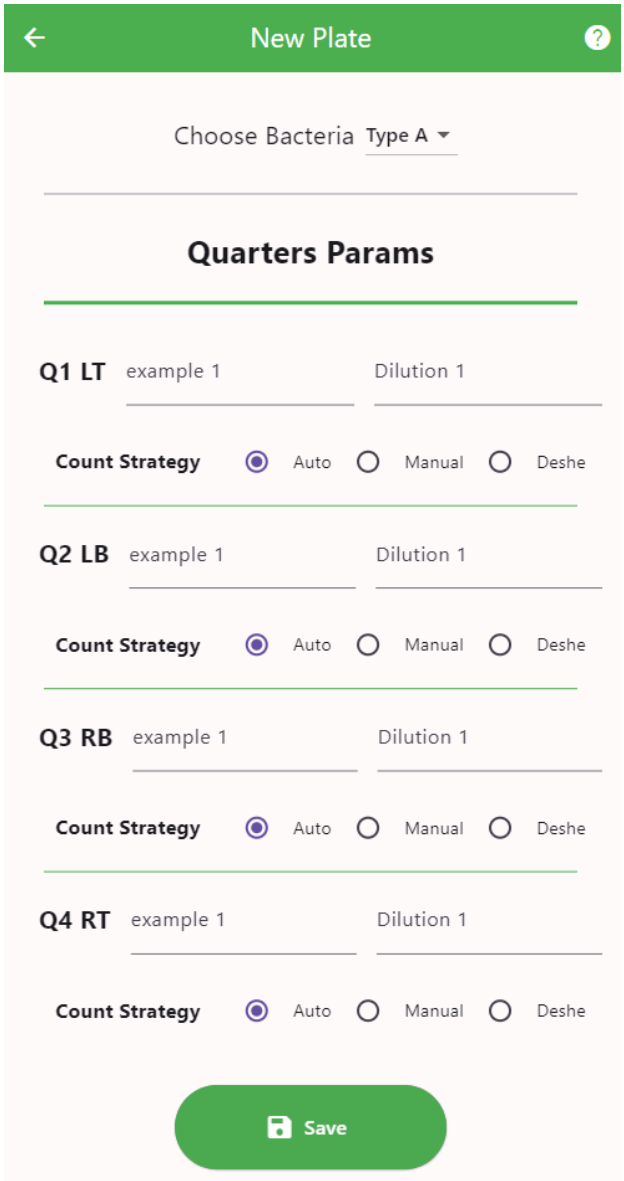


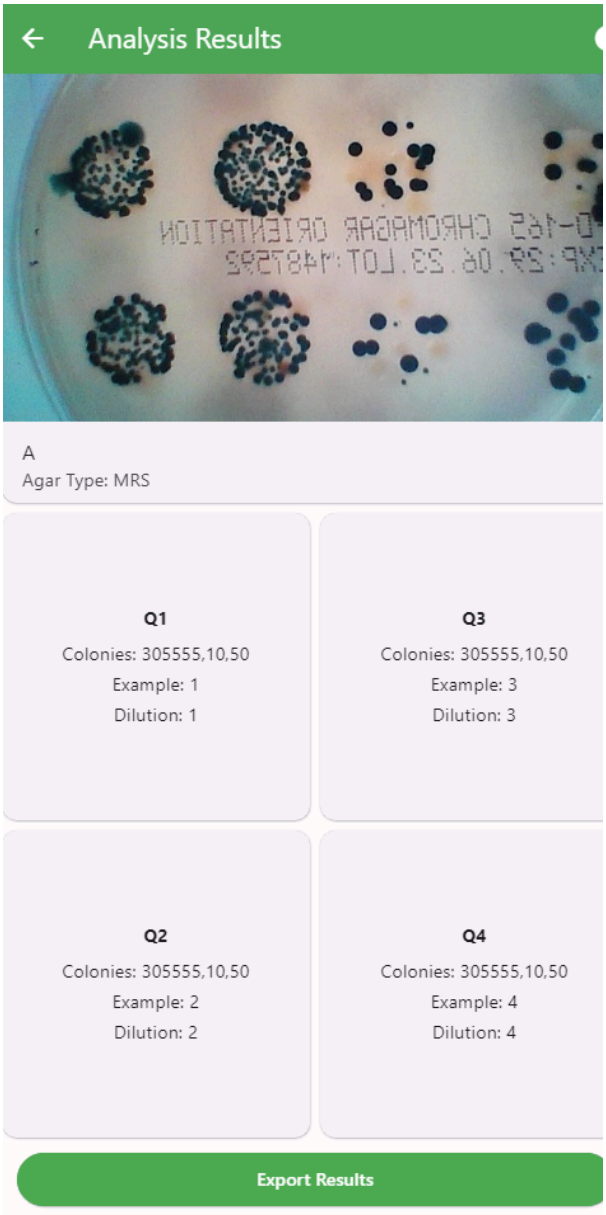
4. **Capturing or Uploading Plate Images:**
   - Take a photo of the agar plate using the camera or upload an image from your device.
   - Ensure the image is clear and the agar plate is well-lit.
   - Make sure you're using a solid background, preferably black.



5. **Setting Plate Parameters:**
   - After uploading the image, set the relevant plate parameters:
   - Select bacteria type for detection.
   - Fill in plate quarters parameters such as example, dilution and counting method.

6. **Analyzing the Image:**
   - Press the 'Save' button to start the analysis.
   - The app will process the image and display the number of detected colonies on the final result screen.

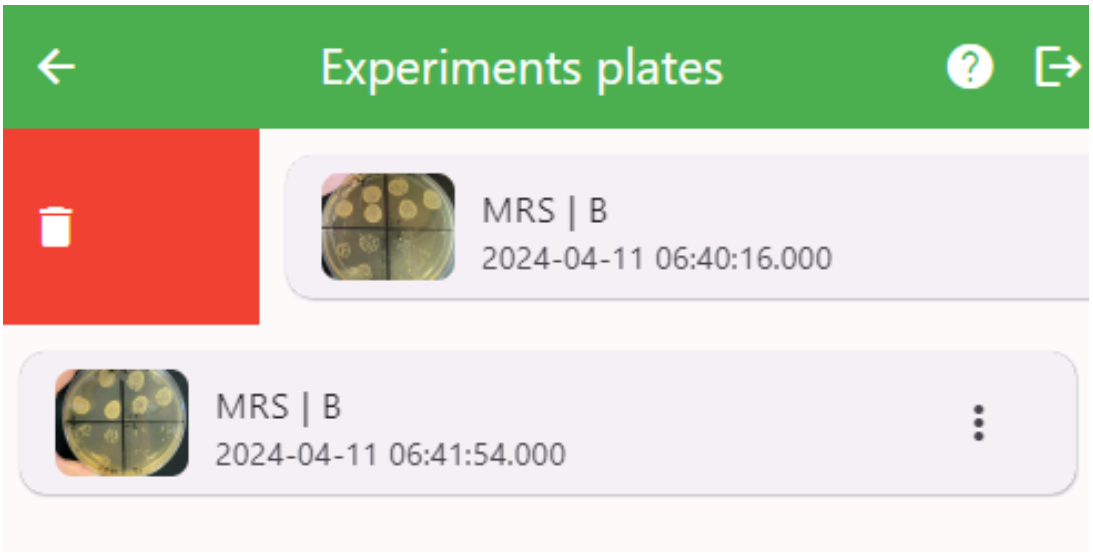




Managing Experiments

1. **Deleting Plates:**

- Swipe the plate you wish to delete to the right and hit 'Delete' to remove the plate.



- Or tap the menu button on the specific plate and hit ‘Delete’ and then confirm deletion.

2. **Reviewing Results:**

- Access the experiment details page to view the results of the analyzed plates.
- Results include the number of colonies detected and specific notes or parameters set for each plate.

Tips for Optimal Use

- Ensure your agar plates are evenly illuminated when capturing images to improve detection accuracy.
- Since the app is a web app, updates are automatic.
- Utilize the app's help and support feature for technical assistance or to provide feedback.
- A Help Guide is present on the top right of the navigation bar in every screen to assist the user.

11. Maintenance Guide

Comprehensive Developer Guide for AgarVision App

This guide is designed to assist developers in taking over, maintaining, and advancing the AgarVision app. It includes setup instructions, environment configuration, building, deployment, and maintenance processes.

System Requirements

Hardware Requirements

- A computer capable of running the latest version of Flutter and its dependencies.
- A computer capable of running the latest version of Python/Flask and its dependencies.
- A computer connected to the network.
- A compatible Android or iOS device for testing, or an emulator/simulator installed on the computer.

Software Requirements

- Flutter SDK: Latest stable version.
- Android Studio or IntelliJ IDEA: For Flutter development.
- Xcode: For iOS development (MacOS only).
- Git: For version control.
- Firebase: Account for backend services.
- VS Code: IDE for backend development.
- Python 3: for backend.
- Docker Desktop installed and running. (for building docker images and deploying the app).

Setting Up the Development Environment

Flutter Installation

1. Download and install the Flutter SDK from the [Flutter official website](https://flutter.dev/docs/get-started/install).
2. Add Flutter to the system path.

IDE Setup

1. Install Android Studio or IntelliJ IDEA.
2. Install Flutter and Dart plugins from the IDE's plugin marketplace.
3. Configure the Flutter SDK path in the IDE settings.

Clone the Repository

```
git clone https://github.com/IbraheemGanayim/AgarVision.git
cd AgarVision
```

Dependency Installation

```
flutter pub get
```

Firebase Setup

- Configure Firebase for Flask server as per Firebase documentation, including downloading the respective `google-services.json` and `GoogleService-Info.plist` files.

Building and Running the App

Android: Open the project in Android Studio, select a device or emulator, and hit the Run button.
iOS: Open `ios/Runner.xcworkspace` in Xcode, select an iOS simulator or connected device and press the Run button.
Web: Open the project in Android Studio, select “Chrome”, and hit the Run button.

Testing

- Navigate to the `test` directory to view or add new tests.
- Execute tests using:

  ```
  flutter test
  ```

Building for Production

Flutter - Client

Running docker build as per docker documentation to build a docker image for the web. A Dockerfile exists in the project that includes building the app in a web platform and creating a Docker image from it.

Flask - Server

Running docker build as per docker documentation to build a docker image for the web. A Dockerfile exists in the project that includes building the app and creating a Docker image from it.

Deployment to Heroku

1. **Run:** heroku login
2. **Run:** heroku container:login
3. **Run:** deploy.bat

   This batch file will build the docker image and deploy it to Heroku.

   Each project has a **deploy.bat** file.

Maintenance

- Regularly update Flutter SDK and packages.
- Regularly update Backend dependencies, for example, Flask and Sklearn.
- Monitor app performance and manage Firebase configurations.

Documentation

Continuously update `README.md` and other documentation files with any significant changes.

This guide is structured to provide a comprehensive overview for developers to effectively manage and enhance the AgarVision app. By following these steps, developers can ensure smooth transitions and ongoing improvements to the app's functionality and performance.