**Classification of Fresh and Rotten Fruits using Computer Vision**

**Design Document**

**Version 1.0**

****

**Group Id: S2402F4B7E (BC200418293 & MC220201538)**

**Supervisor Name: Zaid Ismail**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 11/07/2024 | 1.0 | This version introduces the Fruit Vision project, an automated classification system designed to distinguish between fresh and rotten fruits using computer vision and deep learning techniques. It includes an overview of the project's objectives, the methodology to be used, and the primary components such as the image Preprocessing module, the CNN-based classification module, and the user interface module. The initial scope, system architecture, and functional and non-functional requirements are also outlined. | **(BC200418293 & MC220201538)** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. [Introduction of Design Document 4](#One)
2. [Research Methodology/Proposed Model 4](#Two)
3. [Sequence Diagrams 5](#Three)
4. [Architecture Design Diagram 7](#Four)
5. [Interface Design 8](#interfacedesign)
6. [Test Cases 10](#Six)
7. [Results and Discussion 18](#Seven)

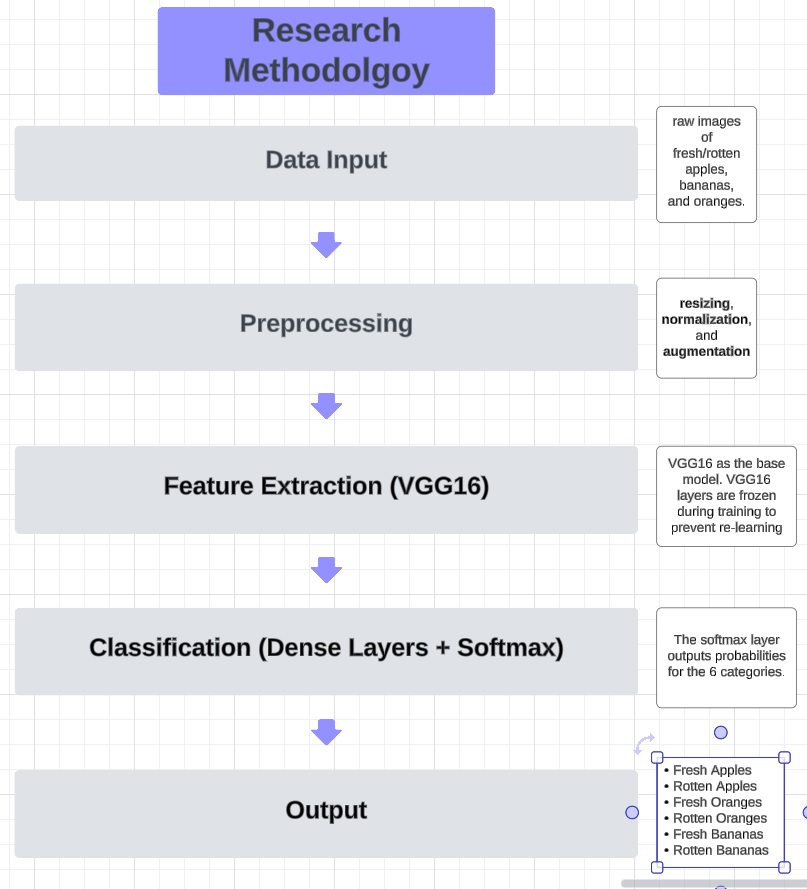
1. [](#Contents)Introduction of Design Document

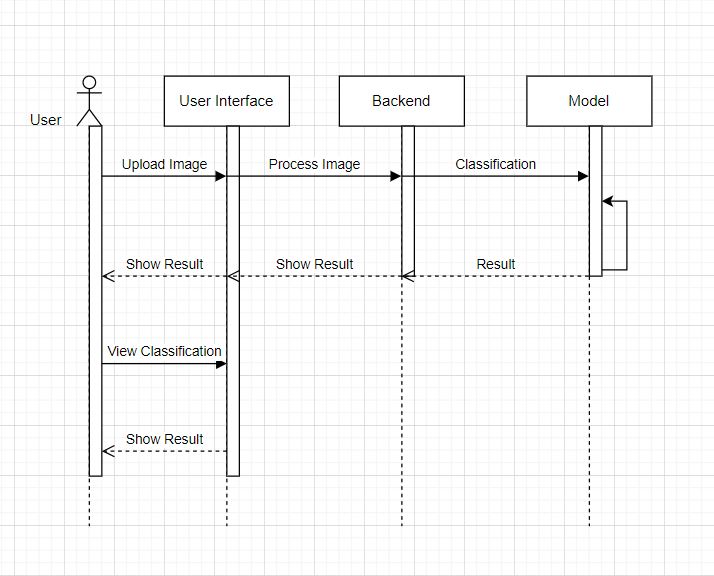
[](#Contents)This document provides a detailed design for the Fruit Vision system, an automated classification system for fresh and rotten fruits using computer vision and deep learning techniques. It includes system architecture, component design, data flow, and user interactions. The purpose of this phase in software development is to create a blueprint for the system that guides the implementation, ensures all requirements are met, and facilitates communication among stakeholders.

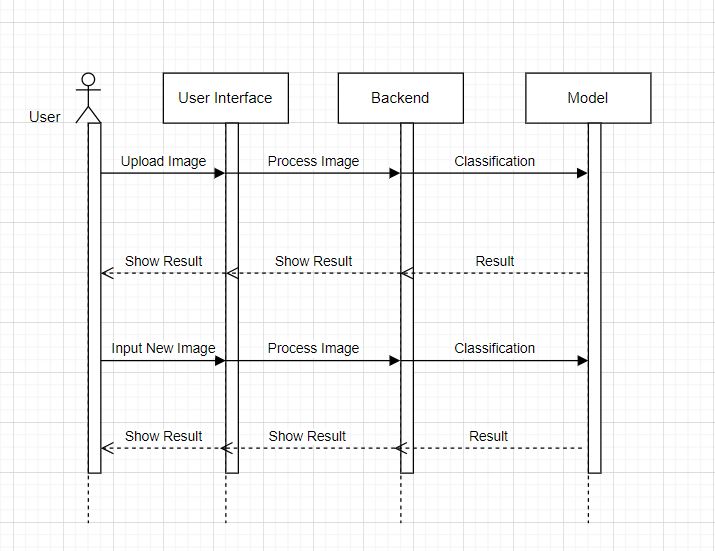
1. Research Methodology/Proposed Model

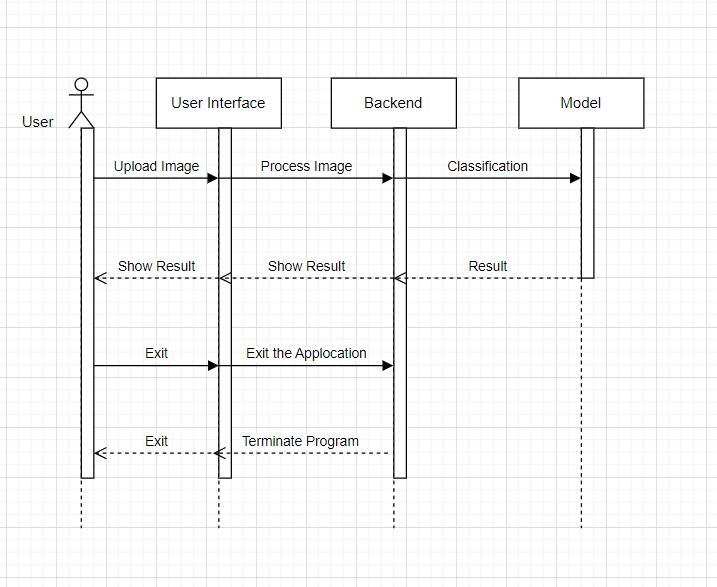
The Fruit Vision system uses a Convolutional Neural Network (CNN) for classifying fruits as fresh or rotten. The research methodology includes:

* **Data Collection**: Using the Kaggle data set of fresh and rotten fruits.
* **Data Preprocessing**: Image cropping, re sizing, and normalization.
* **Model Selection**: Using VGG16 as the base model with modifications for binary classification.
* **Training and Evaluation**: Training the model on the preprocessed data set and evaluating performance using metrics like accuracy, precision, recall, and F1-score.
* **Deployment**: Implementing the model in a user-friendly interface for real-time classification.



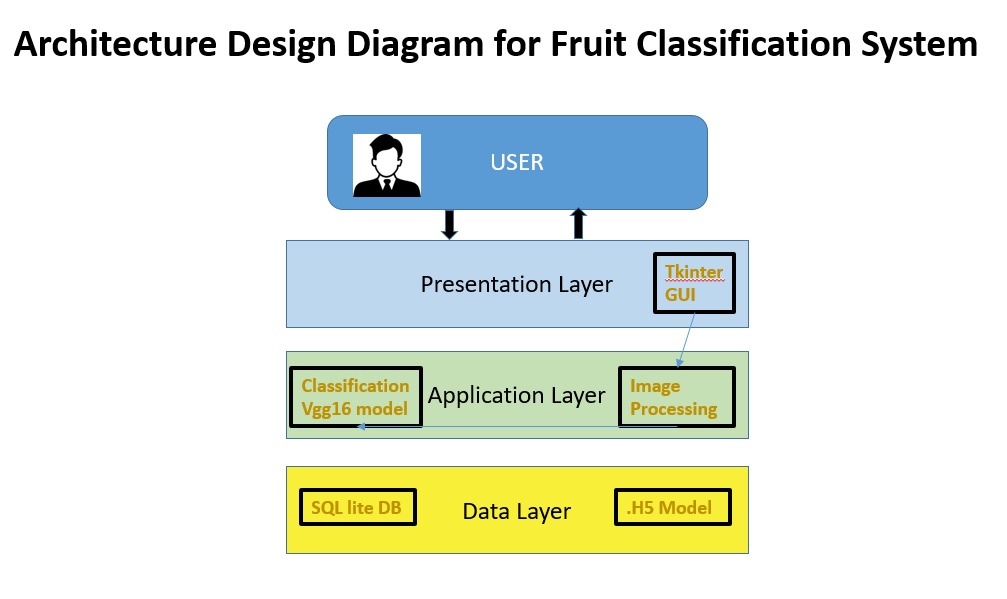
1. [](#Contents)Sequence Diagrams 

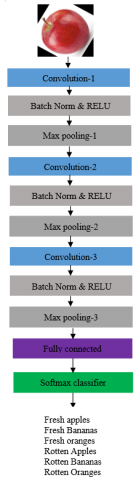




[](#Contents)

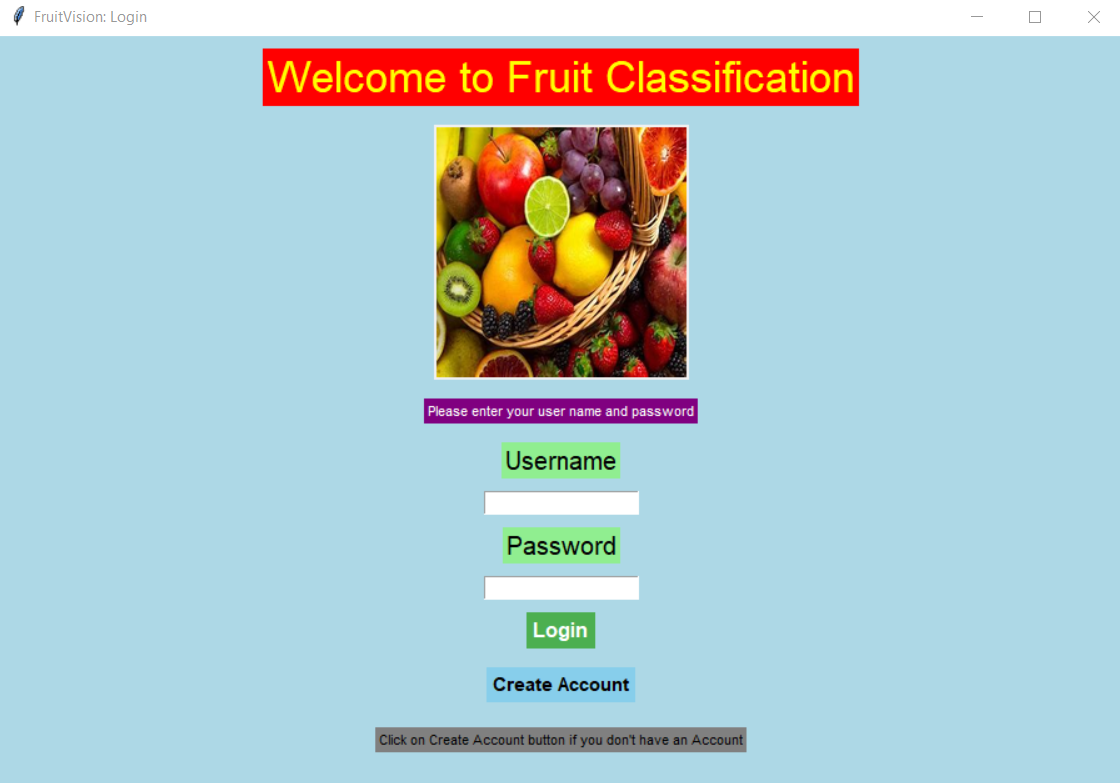
1. Architecture Design Diagram

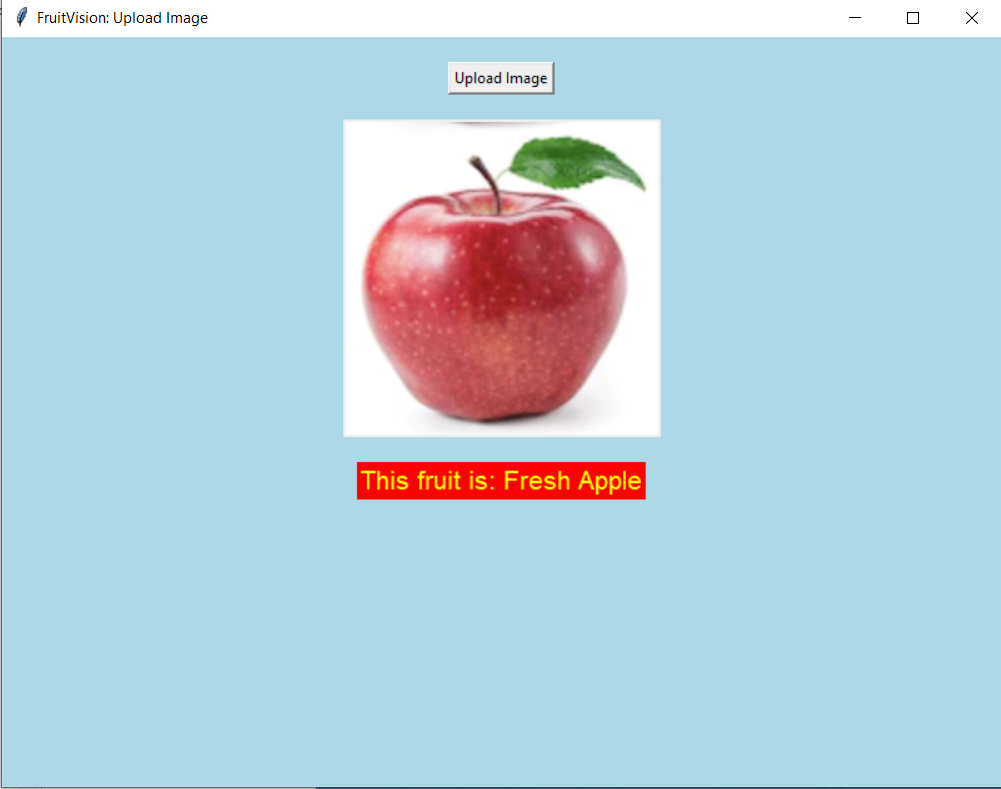




[](#Contents)

1. Interface Design





1. [](#Contents)Test Cases

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **01** | | |
| **Test Case Title** | **Input Image** | | |
| **Actor:** | **(BC200418293 & MC220201538)** | | |
| **Description:** This scenario allows the user to add images to the system. | | | |
| **Preconditions:** The user must have login access for this scenario. | | | |
| **Actions** | | | **Exceptions** |
| 1. The user will log in to the application. | | | “Something went wrong, Please try again”, a message will be shown to the user. |
| 2. User will click the “Add Image” button. | | |
| 3. The user will select the images. | | |
| 4. The system will take the image. | | |
| 5. Use case ends. | | |
| **Post Conditions:**  Success: The users will successfully add images to the system.  Failure: The user cannot add images to the system. | | | |
| **Alternative Paths:** If there is something wrong, the user will make add another image to the system. | | | |
| **Tested by** | |  | |
| **Result** | | **Pass** | |
| **Modification History:** 1.0 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case ID** | **02** | | |
| **Test Case Title** | **Classification Of Image** | | |
| **Actor:** | Model | | |
| **Description:** This scenario allows the user to add images to the system and predict it whether it’s a rotten or a fresh fruit. | | | |
| **Preconditions:** The user must login to the system for this scenario. | | | |
| **Actions** | | | **Exceptions** |
| 1. The user will log in to the application. | | | “Something went wrong, Please try again”, a message will be shown to the user. |
| 2. User will click the “Add Image” button. | | |
| 3. The user will select the images. | | |
| 4. After selecting the image user will predict whether it’s a fresh or rotten fruit. | | |
| 5. Use case ends. | | |
| **Post Conditions:**  Success: The users will successfully add images to the system and predict it.  Failure: The user cannot add images to the system. | | | |
| **Alternative Paths:** If there is something wrong, the user will make add another image to the system. | | | |
| **Tested by** | |  | |
| **Result** | | **Pass** | |
| **Modification History:** 1.0 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case ID** | **03** | | |
| **Test Case Title** | **Output Classification** | | |
| **Actor:** | Model | | |
| **Description:** This scenario allows the user to view images from the system. | | | |
| **Preconditions:** The user must have login access for this scenario. | | | |
| **Actions** | | | **Exceptions** |
| 1. The user will log in to the application. | | | “No record found”, a message will be shown to the user. |
| 2. User will click the “View Image” button. | | |
| 3. The system will show images to the user on the screen. | | |
| 4. The user will see the images with category. | | |
| 5. Use case ends. | | |
| **Post Conditions:**  Success: The users will successfully view images on the screen.  Failure: The user cannot view images on the screen. | | | |
| **Alternative Paths:** If there is something wrong, the user will make view another image. | | | |
| **Tested by** | |  | |
| **Result** | | **Pass** | |
| **Modification History:** 1.0 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case ID** | **04** | | |
| **Test Case Title** | **View Classification** | | |
| **Actor:** | User | | |
| **Description:** This scenario allows the user to monitor the processing of the system. | | | |
| **Preconditions:** The user must have login access for this scenario. | | | |
| **Actions** | | | **Exceptions** |
| 1. The user will log in to the application. | | | “Something went wrong, Please try again”, a message will be shown to the user. |
| 2. Give some instructions to the system. | | |
| 3. The system will start its processing. | | |
| 4. The user will monitor the work processing. | | |
| 5. Use case ends. | | |
| **Post Conditions:**  Success: The users will successfully monitor the system processing.  Failure: The user cannot monitor the system’s processing. | | | |
| **Alternative Paths:** If there is something wrong, the user can go back to the main page and again start monitoring the system. | | | |
| **Tested by** | |  | |
| **Result** | | **Pass** | |
| **Modification History:** 1.0 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case ID** | **05** | | |
| **Test Case Title** | **Input New Image** | | |
| **Actor:** | **(BC200418293 & MC220201538)** | | |
| **Description:** In this scenario, the system will Import the image dataset of fresh and rotten fruit from the link | | | |
| **Preconditions:** There must be some link added to the system from which the system will import the image. | | | |
| **Actions** | | | **Exceptions** |
| 1. The system will check the link. | | | “There is nothing found, Please add some link to the system”, a message will be shown to the user. |
| 2. System will import images by that link. | | |
| 3. The system will start processing according to the instructions. | | |
| 5. Use case ends. | | |
| **Post Conditions:**  Success: The system will work correctly.  Failure: An exception will be shown on the screen. | | | |
| **Tested by** | |  | |
| **Result** | | **Pass** | |
| **Modification History:** 1.0 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case ID** | **06** | | |
| **Test Case Title** | **Exit** | | |
| **Actor:** | User | | |
| **Description:** Actor can Logout the system | | | |
| **Pre-Conditions:** The actor is logged in to the Application before logging out. | | | |
| **Actions** | | | **Exceptions** |
| 1. The Actor selects the Logout button accessible from all system pages. | | | The user cannot log out if he is not logged in or registered already. |
| 2. The Actor clicks on the logout button the system displays a message requesting the user to confirm the logout. | | |
| 3. The Actor selects the Confirm button. | | |
| 4. The system displays the Home page. | | |
| 5. Use case ends. | | |
| **Post Conditions:**  The Actor will log out and return to the login page. | | | |
| **Alternate Paths:** The Admin will return to its Admin panel and the user will return to Main Page. | | | |
| **Tested by** | |  | |
| **Result** | | **Pass** | |
| **Modification History:** 1.0 | | | |

1. [](#Contents)Results and Discussion

**Results Table:**

Classification Report:

precision recall f1-score support

freshapples 0.99 0.98 0.98 395

freshbanana 0.99 1.00 1.00 381

freshoranges 1.00 0.98 0.99 388

rottenapples 0.97 1.00 0.98 601

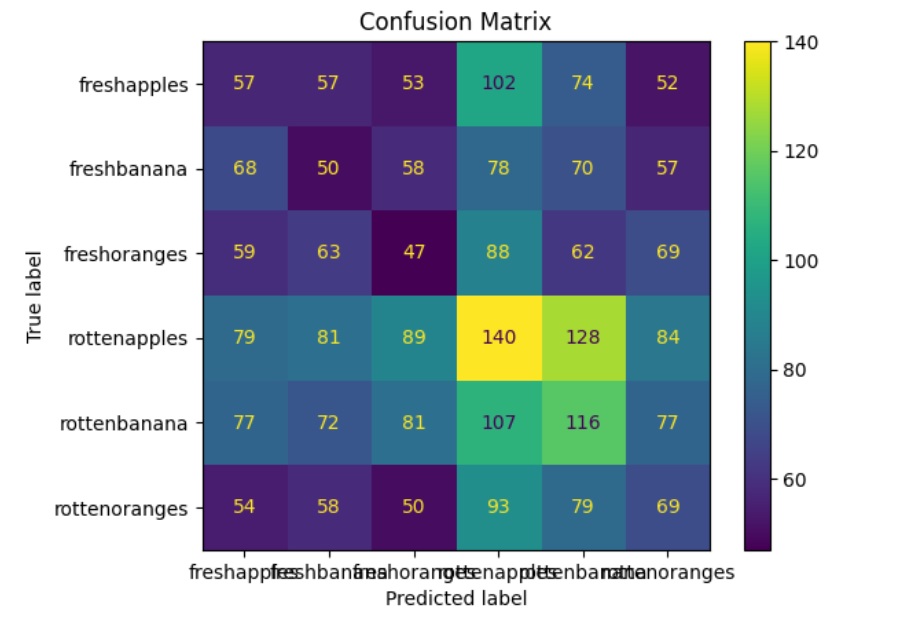
rottenbanana 1.00 1.00 1.00 530

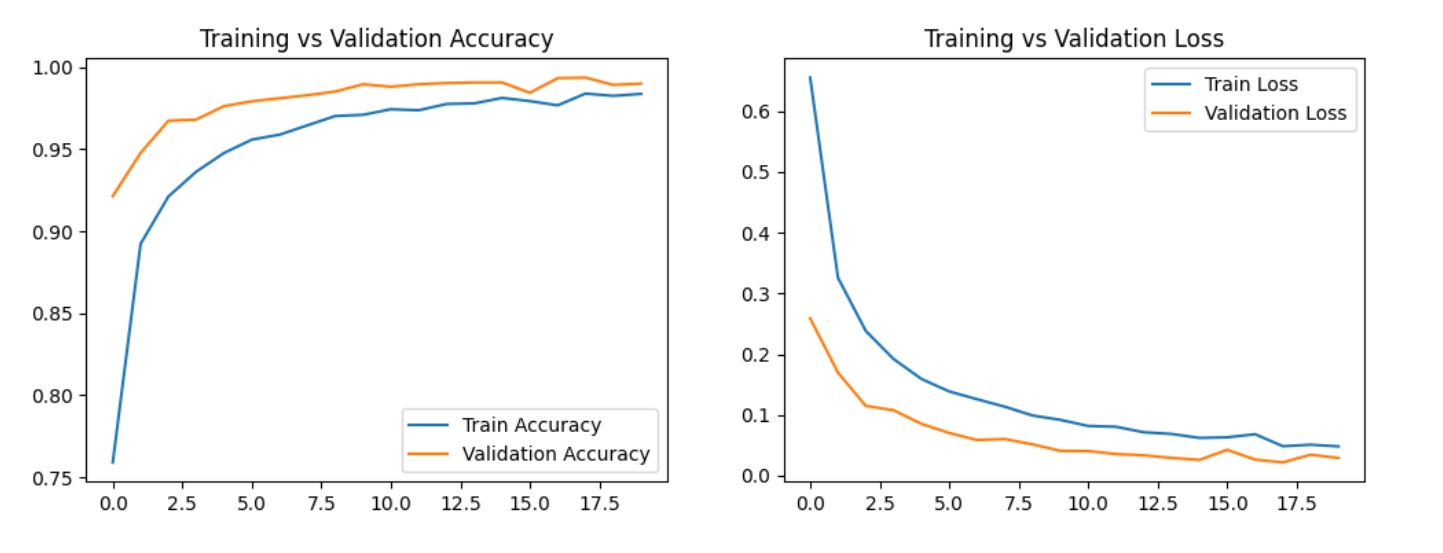
rottenoranges 0.99 0.98 0.99 403

accuracy 0.99 2698

macro avg 0.99 0.99 0.99 2698

weighted avg 0.99 0.99 0.99 2698

****

****

**Discussion:**

Discuss the performance of each model:

* **VGG16**: High accuracy and balanced precision/recall indicate robust performance in classifying fresh and rotten fruits.
* **CustomCNN**: Slightly lower performance but faster inference time, making it suitable for real-time applications.