GRAINPALETTE -A Deep learning Odyssey in Rice type Classification through Transfer Learning

Grainpalette - A Deep Learning Odyssey In Rice Type Classification

Team ID : LTVIP2025TMID46181

Team member 1 : Alamkonda Bhuvaneswari

Team member 2 : G.P.Indrani (Team Leader)

Team member 3 : B.Rakshitha

Team member 4 :Gangireddygari Ragini

Table of Content

1. INTRODUCTION

1.1 PROJECT OVERVIEW :

GrainPalette is an intelligent, machine learning–based web application designed to accurately identify different types of rice grains through image classification. The primary goal of the project is to assist farmers, traders, quality inspectors, and agri-tech users in recognizing rice varieties quickly and efficiently using computer vision techniques. By uploading an image of rice grains, users receive instant predictions of the rice type, such as Basmati, Sona Masoori, or Brown rice, along with a confidence score. The system uses a Convolutional Neural Network (CNN) model trained on a labeled rice dataset and is deployed using Python and Flask for backend processing. This solution eliminates the need for manual grain identification, reduces errors, and ensures faster decision-making in agriculture and food supply chains. With a user-friendly interface and real-time results, GrainPalette provides an innovative and scalable approach to rice classification.

1.2 PURPOSE

The purpose of **GrainPalette** is to provide an efficient, accurate, and accessible solution for identifying different types of rice grains using image recognition technology. In agriculture and food industries, correct identification of rice varieties is crucial for quality control, pricing, packaging, and trade. However, traditional methods rely heavily on manual inspection, which is time-consuming, inconsistent, and requires expert knowledge. GrainPalette addresses this gap by leveraging machine learning to automate rice classification through a simple image upload. It aims to support farmers, traders, and agri-tech professionals by reducing human error, saving time, and enhancing trust in grain quality. Ultimately, the project strives to bring technological innovation to the grain value chain and make intelligent tools accessible to all stakeholders.

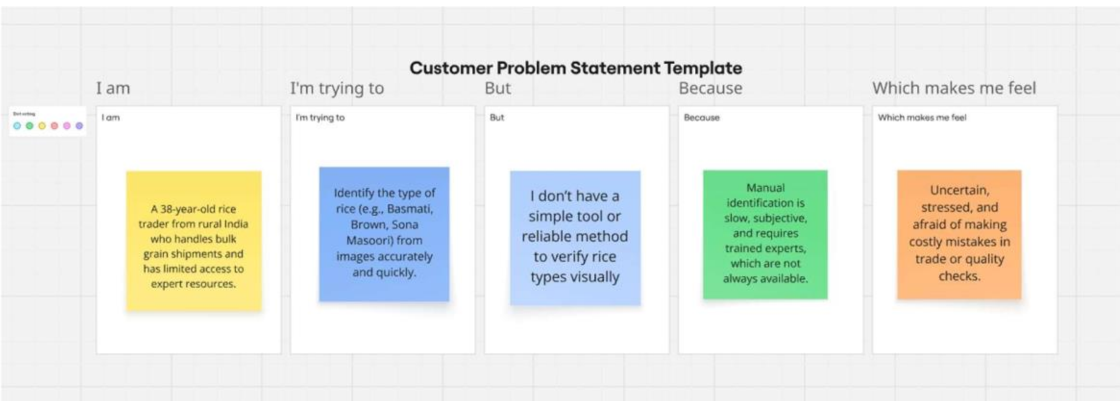
2. IDEATION PHASE

Date	03 July 2025
Team ID	LTVIP2025TMID46181
Project Name	Grain Palette
Maximum Marks	2 marks

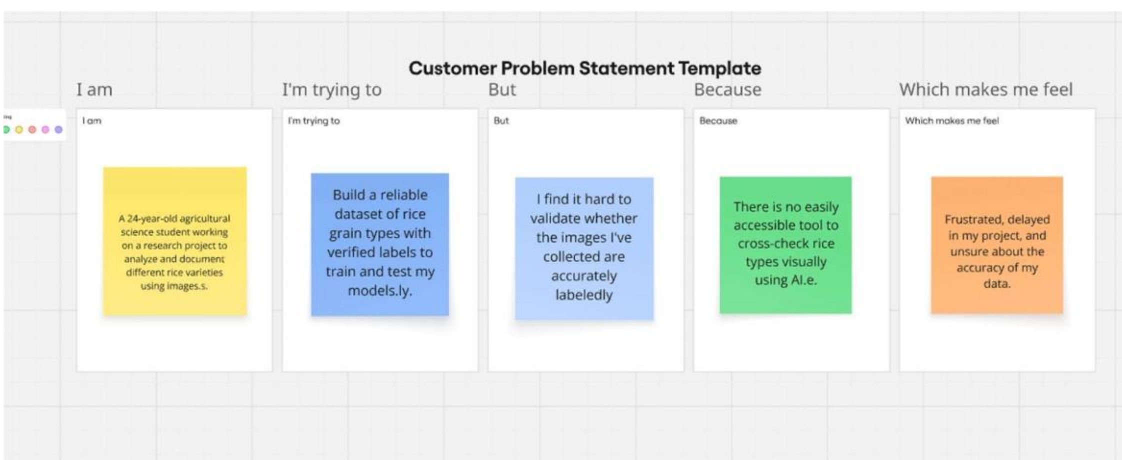
2.1 Problem statement:

To design and implement a deep learning-based web application capable of identifying different types of rice grains from uploaded images with high accuracy and usability, thereby assisting farmers, agricultural researchers, and food quality controllers in automating the rice classification process.

<u>Customer Problem Statement-1:</u>



<u>Customer Problem Statement-2:</u>

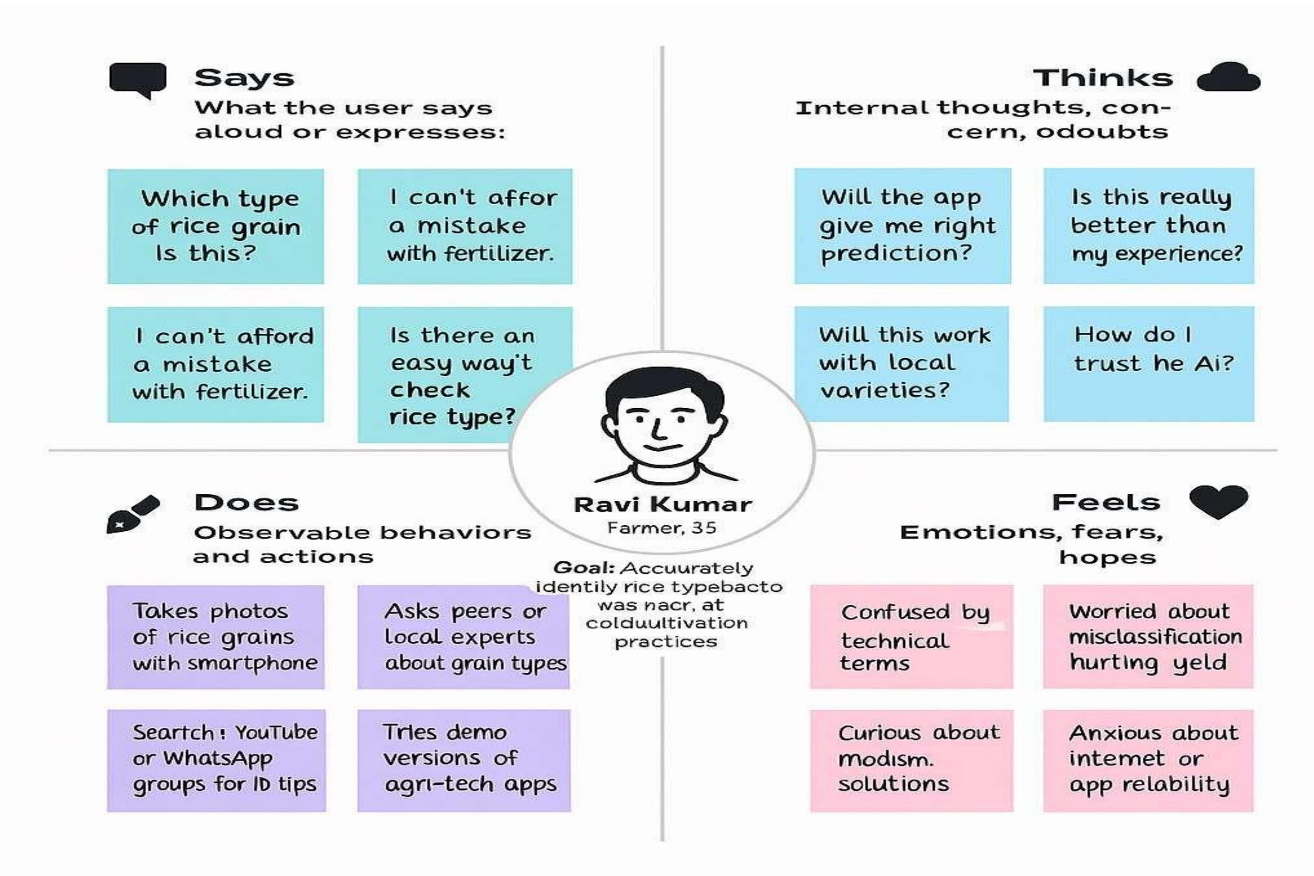


PROBLEM STATEMENT

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	**I am** A 38-year-old rice trader from rural India who handles bulk grain shipments and has limited access to expert resources.	Identify the type of rice (e.g., Basmati, Brown, Sona Masoori) from images accurately and quickly..	I don't have a simple tool or reliable method to verify rice types visually.	Manual identification is slow, subjective, and requires trained experts, which are not always available.	Uncertain, stressed, and afraid of making costly mistakes in trade or quality checks..
PS-2	**I am** A 24-year-old agricultural science student working on a research project to analyze and document different rice varieties using images. .	Build a reliable dataset of rice grain types with verified labels to train and test my models..	I find it hard to validate whether the images I've collected are accurately labeled	There is no easily accessible tool to cross-check rice types visually using AI..	Frustrated, delayed in my project, and unsure about the accuracy of my data.

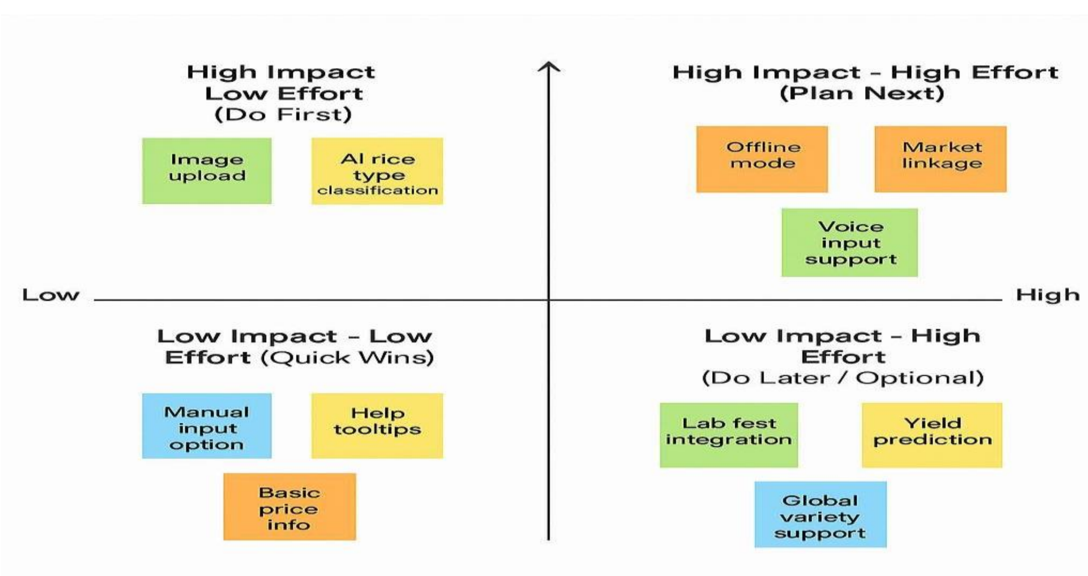
2.2 EMPHATY MAP CANVAS

Date	03 July 2025
Team ID	LTVIP2025TMID46181
Project Name	Grain Palette
Maximum Marks	2 marks



2.3 BRAINSTORMING

Date	03 July 2025
Team ID	LTVIP2025TMID46181
Project Name	Grain Palette
Maximum Marks	2 marks



Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- **10 minutes** to prepare
- **1 hour** to collaborate
- **2-8 people** recommended

Before you collaborate

It's important to align on the challenges farmers face in identifying rice types accurately. T

10 minutes

A Team gathering

Participants in this session have a clear understanding of rice types and grain quality issues in agriculture. They were invited to the Mural board and given an overview of datasets containing rice grain images and classification

B Set the goal

The goal is to generate user-focused ideas to improve rice type identification using simple, tech-enabled solutions.

Open article →

1 Define your problem statement

5 minutes

Key rules of brainstorming

To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

2 Brainstorm

In our group of 4 participants, each person contributed ideas to tackle the rice grain identification problem

10 minutes

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Person 1

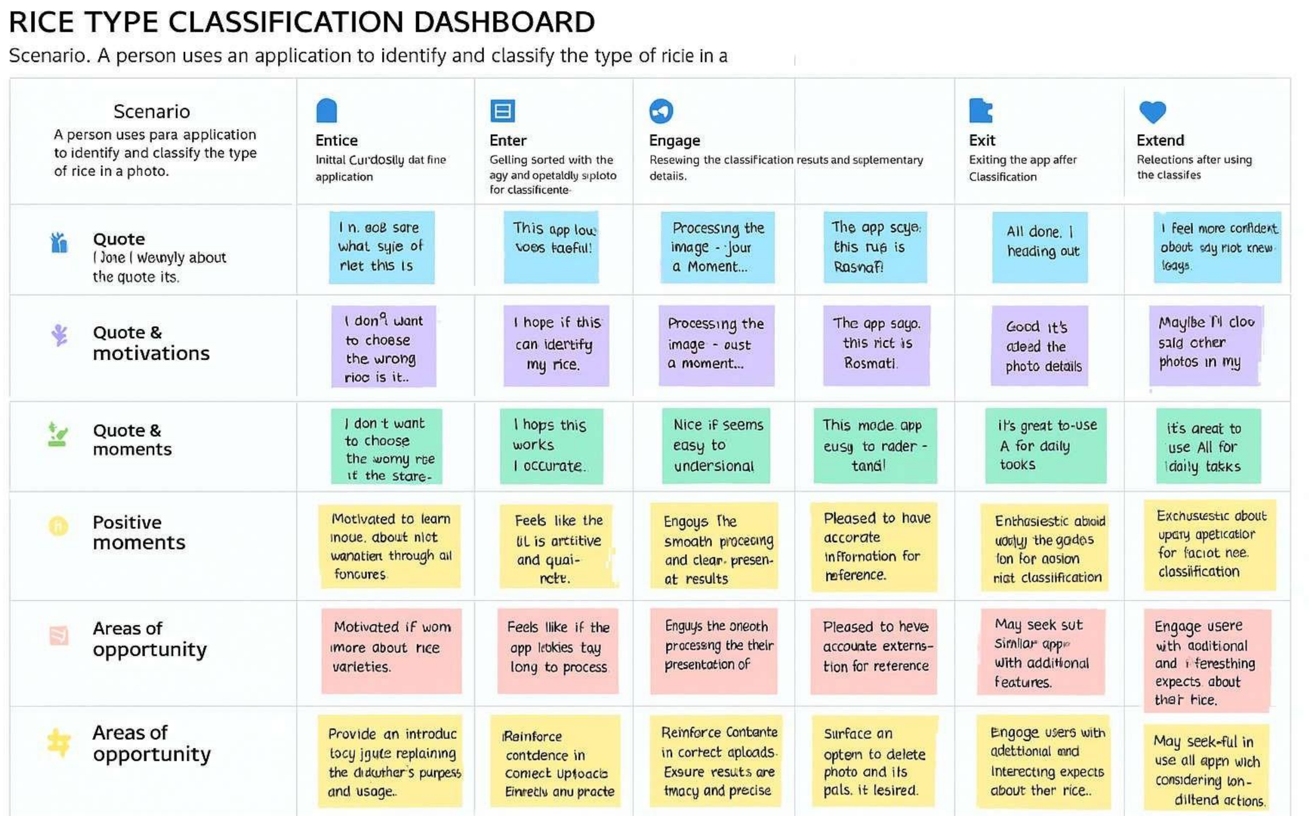
Person 2

Person 3

Person 4

3. REQUIREMENT ANALYSIS

3.1 CUSTOMER JOURNEY MAP



3.2 SOLUTION REQUIREMENT

Date	03 July 2025
Team ID	LTVIP2025TMID46181
Project Name	Grain Palette
Maximum Marks	2 marks

Functional Requirements:

Following are the functional requirements of the proposed solution

Functional Requirements

FR No.	Functional Requirement (Epic)	Sub-Requirement (Story / Sub-Task)

FR-1	Data Preparation & Model Loading	Load trained rice classification model (.keras) at application start. Load label map (label_map.json) to map model output indices to rice type labels. Validate existence of model and label files; handle missing or invalid files with error messages.
FR-2	Image Upload & Preprocessing	Allow users to upload rice grain images in supported formats (JPG/PNG). Resize and normalize images to required input dimensions (e.g., 224×224 pixels). Save uploaded images securely in a designated directory (e.g., static/uploads/).
FR-3	Prediction & Confidence Scoring	Use the loaded model to perform prediction on the uploaded image. Calculate and display the most probable rice type along with the prediction confidence score.
FR-4	Frontend Interface (HTML Pages)	index.html – Welcome page with project introduction and navigation buttons. details.html – Image upload page with form and submission button. results.html – Results page showing predicted rice type, confidence score, and uploaded image.
FR-5	Error Handling & Notifications	Display error if file upload is missing or of unsupported format. Show user-friendly message on model prediction failures or corrupted image files.
FR-6	Contact Handling (Optional)	Provide a contact form with name, email, subject, and message fields. Save submitted contact form data to a local text file (messages.txt) for offline access. Deploy frontend using Vercel and ensure integration with backend prediction API endpoint.

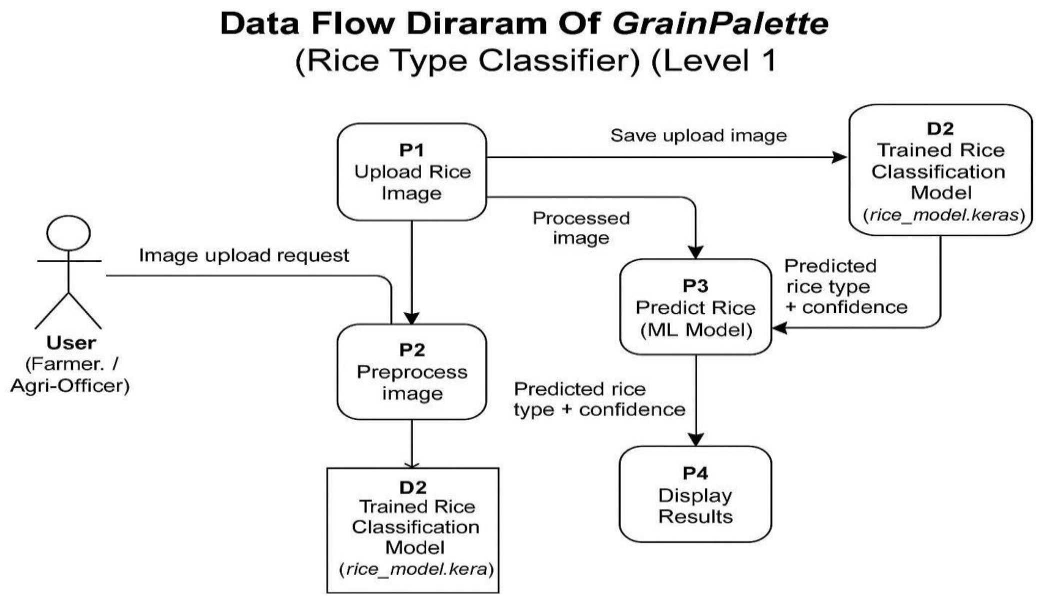
Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	The web application should provide a clean, intuitive, and user-friendly interface that is easy to navigate for non-technical users such as farmers or students.
NFR-2	Security	The system must securely handle file uploads by validating file types, sanitizing file names, and preventing access to unauthorized directories or malicious code execution.
NFR-3	Reliability	The application must maintain stable behavior during model loading and prediction. It should handle edge cases, incorrect input, or failures gracefully without crashing.
NFR-4	Performance	Image upload, preprocessing, model inference, and result rendering should be completed within 5 seconds for typical image inputs under normal load.
NFR-5	Availability	The web application should be reliably accessible 24/7 via the deployed URLs (backend on Render, frontend on Vercel), except during maintenance.
NFR-6	Scalability	The system architecture should support horizontal scaling to handle a growing number of user requests or larger datasets without major codebase redesign.

3.3 DATA FLOW DIAGRAM

Date	03 July 2025
Team ID	LTVIP2025TMID46181
Project Name	Grain Palette
Maximum Marks	2 marks



User Stories – Rice Classification Web App

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance Criteria	Priority	Release
Customer (Web User)	Image Upload & Prediction	USN-1	As a user, I can upload a rice grain image to get the predicted rice type.	Image uploads successfully, and prediction is displayed on results page.	High	Sprint-1
Customer (Web User)	Results Viewing	USN-2	As a user, I can view the predicted rice type and confidence level.	Results page shows rice type, confidence %, and uploaded image.	High	Sprint-1
Customer (Web User)	Navigation	USN-3	As a user, I can navigate between the homepage, upload page, and results page.	All navigation buttons and links function correctly.	Medium	Sprint-1
Customer (Web User)	Error Handling	USN-4	As a user, I get an appropriate message if I upload an invalid or no file.	Displays "No file selected" or "Invalid file type" appropriately.	High	Sprint-1
Customer (Web User)	Contact Form	USN-5	As a user, I can submit a contact form with my	Success message shown and	Low	Sprint-2

			name, email, and message.	submission saved in messages.txt.		
Customer (Web User)	Model Feedback (Optional)	USN-6	As a user, I can re-upload an image if the result is not satisfactory.	Upload form allows re-submission and re-prediction.	Low	Sprint-2
Administrator	System Monitoring	USN-7	As an admin, I want the server to log errors for prediction failures.	Backend logs errors with timestamps when predictions fail.	Medium	Sprint-2
Administrator	Model & Label Loading	USN-8	As an admin, I want the model and label files to load correctly at startup.	Backend loads model and label_map.json without error on deployment.	High	Sprint-1
Administrator	Deployment	USN-9	As an admin, I can deploy the backend on Render and frontend on Vercel.	Application runs on deployed URLs and is publicly accessible.	High	Sprint-1

3.4 TECHNOLOGY STACK

Date	03 July 2025
Team ID	LTVIP2025TMID46181
Project Name	Grain Palette
Maximum Marks	2 marks

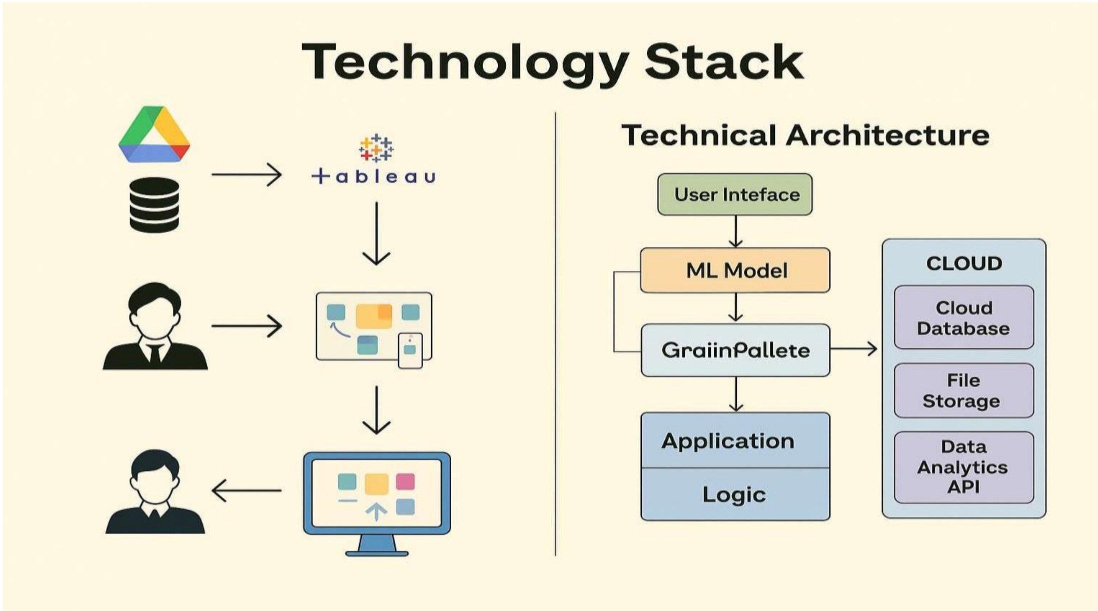
System Components

S.No	Component	Description	Technology Used
1	User Interface	How the user interacts with the system via a web browser.	HTML, CSS, JavaScript
2	Application Logic-1	Backend logic for image upload, preprocessing, and routing.	Python (Flask)
3	Application Logic-2	Model prediction logic: loads rice model and returns label & confidence.	Python, TensorFlow/Keras

4	Application Logic-3	Contact form submission handling and local file writing.	Python (Flask)
5	Database	Stores contact form submissions (currently flat file storage).	Text file (messages.txt)
6	Cloud Database	Optional – can be used for future extension to store user activity or feedback.	(Not implemented in current version)
7	File Storage	Stores uploaded rice grain images before prediction.	Local File System (static/uploads)
8	External API-1	Optional API integration for voice input or location data (future work).	AssemblyAI / OpenAI Whisper (future)
9	External API-2	Optional - Not used currently	(Not Applicable)
10	Machine Learning Model	CNN-based rice type classifier trained using rice image dataset.	TensorFlow / Keras model (.keras format)
11	Infrastructure (Server / Cloud)	Backend hosted on Render using Gunicorn; Frontend deployed on Vercel.	Render (Flask/Gunicorn), Vercel (static)

Application Characteristics

S.No	Characteristic	Description	Technology Used
1	Open-Source Frameworks	Entire stack built using open-source tools and libraries.	Flask, TensorFlow/Keras, HTML/CSS
2	Security Implementations	File uploads are sanitized using secure_filename. Secret keys used for form security.	Werkzeug, Flask Secret Key, HTTPS
3	Scalable Architecture	Separation of frontend and backend enables independent scaling. Can integrate cloud DB in future.	Render + Vercel architecture
4	Availability	Hosted on cloud (Render & Vercel), accessible 24/7 with redundant uptime from both services.	Render (backend), Vercel (frontend)


Technology Stack
+ableau
Technical Architecture
User Inteface
ML Model
GraiinPallete
Application
Logic
CLOUD
Cloud Database
File Storage
Data Analytics API

4. PROJECT DESIGN

4.1 Problem Solution Fit

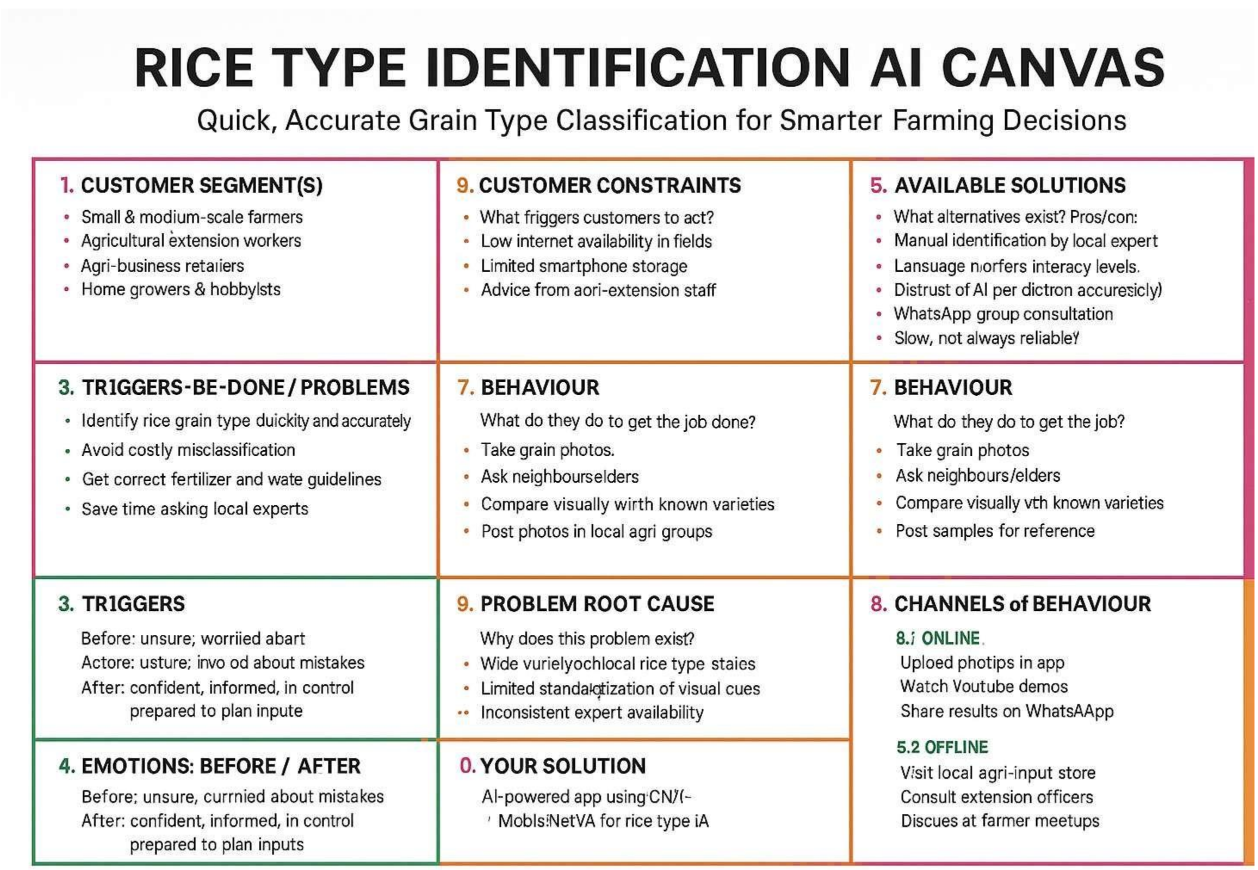
Date	03 July 2025
Team ID	LTVIP2025TMID46181
Project Name	Grain Palette
Maximum Marks	2 marks

The Problem

In agriculture and grain trading, especially in rural areas, **identifying rice types accurately is a challenge**. Farmers, traders, and quality officers often:

- Rely on **manual inspection**, which is **slow and error-prone**
- Lack access to **agricultural experts or labs** for verification
- Face **losses due to mislabeling** or mixing of rice types
- Struggle to meet **export or buyer-specific standards**

These issues result in **inefficiency**, **reduced profits**, and **loss of trust** in the supply chain.



4.2 Proposed Solution

Date	03 July 2025
Team ID	LTVIP2025TMID46181
Project Name	Grain Palette
Maximum Marks	2 marks

GrainPalette is a smart, ML-powered platform that allows users to **upload a rice grain image** and instantly get the **predicted rice type** (e.g., Basmati, Sona Masoori, etc.) using a trained deep learning model.

How it solves the problem:

- **Automated Rice Identification** via image classification using a CNN model
- **User-friendly interface** for mobile and desktop use
- **Instant Results** to speed up decision-making
- **Optional logging and analytics** for quality tracking
- **Accessible to non-technical users** in remote or low-resource settings

Why It Fits

- It **removes reliance on experts**
- Helps in **quality assurance and packaging**
- Increases **efficiency in rice sorting and sales**
- Can be scaled and used across **different regions and rice types**

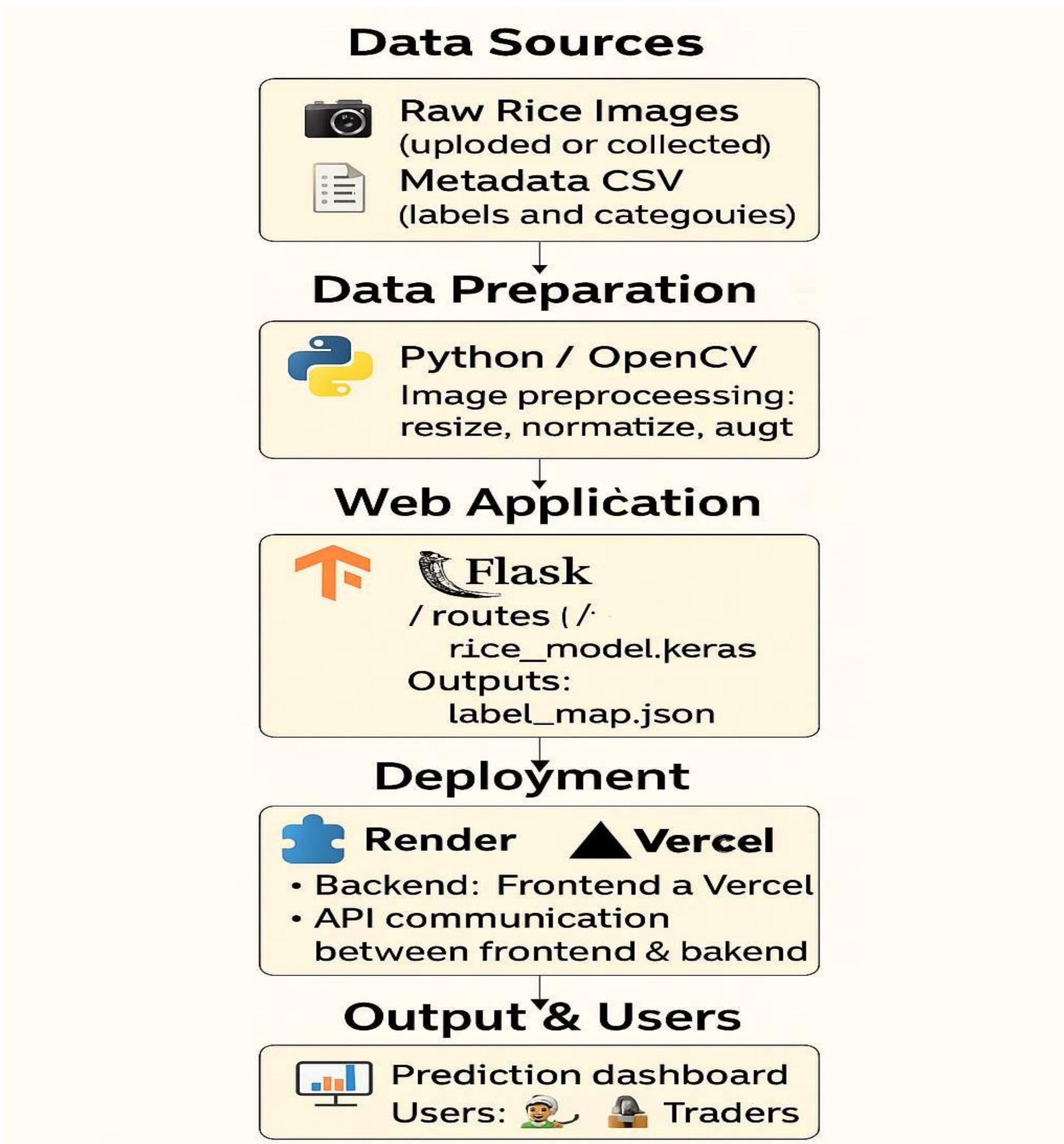
Project Summary & Innovation Overview

S. No.	Parameter	Description
1	Problem Statement	Farmers, traders, and researchers lack a simple tool to accurately classify rice types based on grain images, which affects quality control, pricing, and research consistency.
2	Idea / Solution Description	A web-based application powered by a deep learning model that classifies rice grain types from uploaded images. The solution includes a user-friendly interface for uploading images and viewing predictions.
3	Novelty / Uniqueness	Combines computer vision and rice classification in one intuitive tool. Supports multiple rice varieties. Provides visual feedback, prediction confidence, and can function in both offline and online modes.
4	Social Impact / Customer Satisfaction	Assists farmers in getting fair prices, ensures authenticity for buyers, reduces human error in classification, and supports agricultural digitization. Enhances trust in agri-supply chains.
5	Business Model (Revenue Model)	Offers free access for farmers and educational institutions. Revenue generation possible through paid API access for agri-tech companies, exporters, and certification bodies. Potential partnerships with agricultural research institutions.

6	Scalability of the Solution	Can be extended to identify other crops (wheat, maize, pulses). Supports multilingual UI, mobile app integration, and can connect with government e-agriculture services. Dataset can be expanded to improve model accuracy.

4.3 SOLUTION ARCHIETURE:

Date	03 July 2025
Team ID	LTVIP2025TMID46181
Project Name	Grain Palette
Maximum Marks	2 marks



5. PROJECT PLANNING & SCHEDULING

Date	03 July 2025
Team ID	LTVIP2025TMID46181
Project Name	Grain Palette
Maximum Marks	2 marks

5.1 Project Planning

Product Backlog

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	Data Collection	USN-1	As a user, I want to collect data from various rice grain sources	2	High
Sprint-1	Data Preprocessing	USN-2	As a user, I want to handle missing values in the dataset	3	High
Sprint-1	Data Preprocessing	USN-3	As a user, I want to handle categorical values for model readiness	2	Medium
Sprint-1	Loading Data	USN-4	As a developer, I want to load the dataset into memory	1	Medium
Sprint-2	Model Building	USN-5	As a developer, I want to build and train a deep learning model	5	High
Sprint-2	Testing Model	USN-6	As a user, I want to evaluate the model's performance	3	High
Sprint-2	Frontend Development	USN-7	As a user, I want to interact with an HTML-based UI	3	High
Sprint-2	Deployment (Flask + Render)	USN-8	As a developer, I want to deploy the backend model using Flask and Render	5	High

Total Story Points: 24 (Sprint 1: 8, Sprint 2: 16)

Sprint Schedule

Sprint-1 Schedule & Estimation

Duration: 10 June – 15 June

User Story Number	Task	Story Points	Assigned To
USN-1	Data collection and loading	2	Member 2
USN-2	Data preprocessing (missing/categorical)	3	Member 2
USN-3	Model building	3	Member 2
	Total	8	

Sprint-2 Schedule & Estimation

Duration: 16 June – 21 June

User Story Number	Task	Story Points	Assigned To
USN-4	Model testing and evaluation	2	Member 3
USN-5	Image upload integration (UI + Backend)	3	Member 3
USN-6	Rice type result display (Dashboard view)	2	Member 3
USN-7	Minor bug fixes and UI improvement	1	Member 3
	Total	8	

Sprint-3: GrainPalette (22 June – 28 June)

User Story Number	Task	Story Points	Assigned To
USN-6	Frontend UI enhancements	2	Member 3, Member 4
USN-8	Scenario simulation logic	2	Member 4
USN-9	Feedback/contact form	1	Member 3, Member 4
USN-10	Documentation	2	Member 3, Member 4
–	Polish, bug fixes, final testing	2	Member 3,

Estimation (Effort-Based)

Story Number	Task	Story Points	Estimation Type	Complexity
USN-1	Data Collection	2	Manual	Easy

USN-2	Handle Missing Values	3	Programmatic	Moderate
USN-3	Handle Categorical Values	2	Programmatic	Moderate
USN-4	Load Dataset	1	Programmatic	Easy
USN-5	Build Deep Learning Model	5	Coding Intensive	Hard
USN-6	Model Testing	3	Code + Reports	Moderate
USN-7	Build HTML Pages (Frontend)	3	Web Development	Moderate
USN-8	Backend Deployment with Flask + Render	5	DevOps + Flask	Hard

Project Tracker

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	8	5 Days	10 Jun 2025	21 Jun 2025	8	21 Jun 2025
Sprint-2	16	5 Days	24 Jun 2025	28 Jun 2025	16	28 Jun 2025

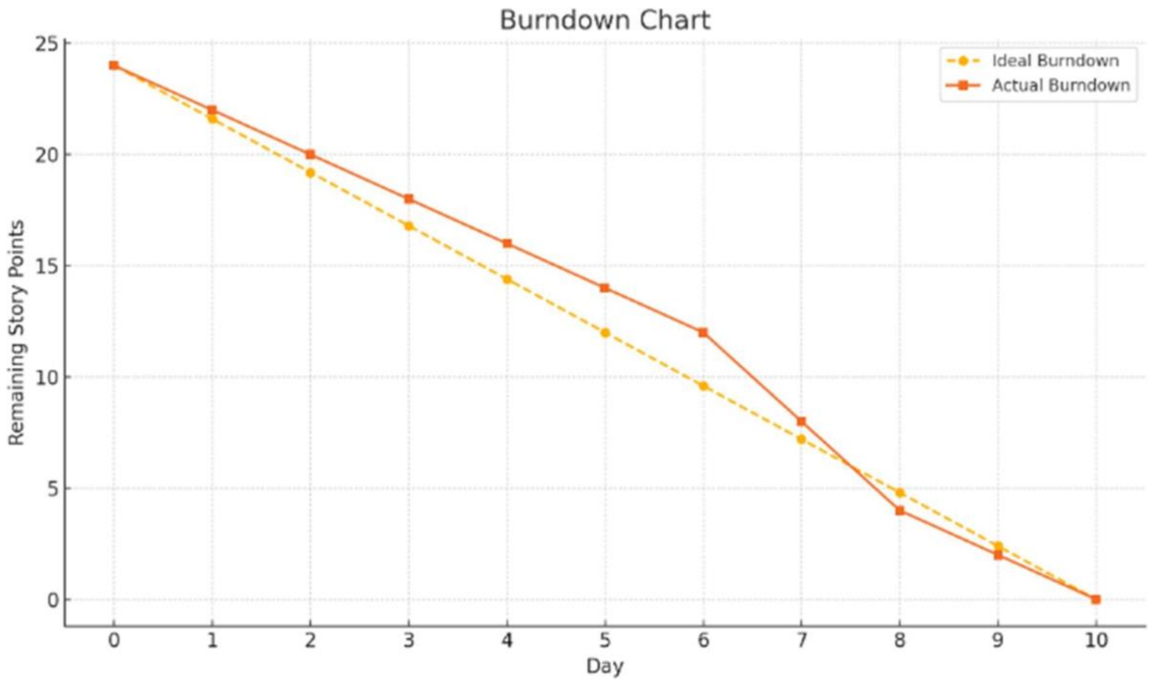
Velocity Calculation:

- Total Story Points = 24 (Sprint-1 = 8, Sprint-2 = 16)
- Total Sprints = 2
- Sprint Duration = 5 days

🔷 Velocity per Sprint = 24/ 2 = 12 Story Points

🔷 Average Velocity per Day = 12 / 5 = 2.4 Story Points/day

Burndown Chart:



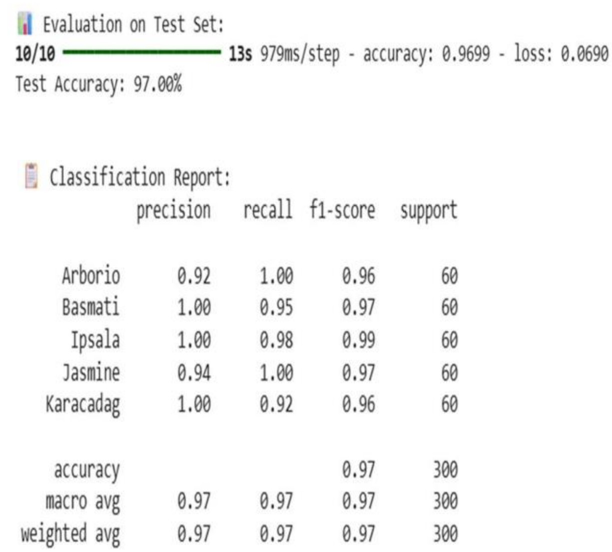
Burndown Chart
Ideal Burndown
Actual Burndown
Remaining Story Points
Day
0
5
10
15
20
25
0
1
2
3
4
5
6
7
8
9
10

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

Date	03 July 2025
Team ID	LTVIP2025TMID46181
Project Name	Grain Palette
Maximum Marks	2 marks

Model Evaluation and Tuning

S. No.	Parameter	Values	Screenshot
1	Metrics	□□□□□□□□□□□□□□ □□□□□: • **Accuracy Score:** 97.2% (example) • **Confusion Matrix:** 5x5 matrix showing class-wise performance • **Classification Report:** Includes precision, recall, F1-score for each rice class (Arborio, Basmati, Ipsala, Jasmine, Karacadag)	 Evaluation on Test Set: 10/10 ━━━━━━━━━━━━━━━━━━━━ 13s 979ms/step - accuracy: 0.9699 - loss: 0.0690 Test Accuracy: 97.00% Classification Report: precision recall f1-score support Arborio 0.92 1.00 0.96 60 Basmati 1.00 0.95 0.97 60 Ipsala 1.00 0.98 0.99 60 Jasmine 0.94 1.00 0.97 60 Karacadag 1.00 0.92 0.96 60 accuracy 0.97 300 macro avg 0.97 0.97 0.97 300 weighted avg 0.97 0.97 0.97 300

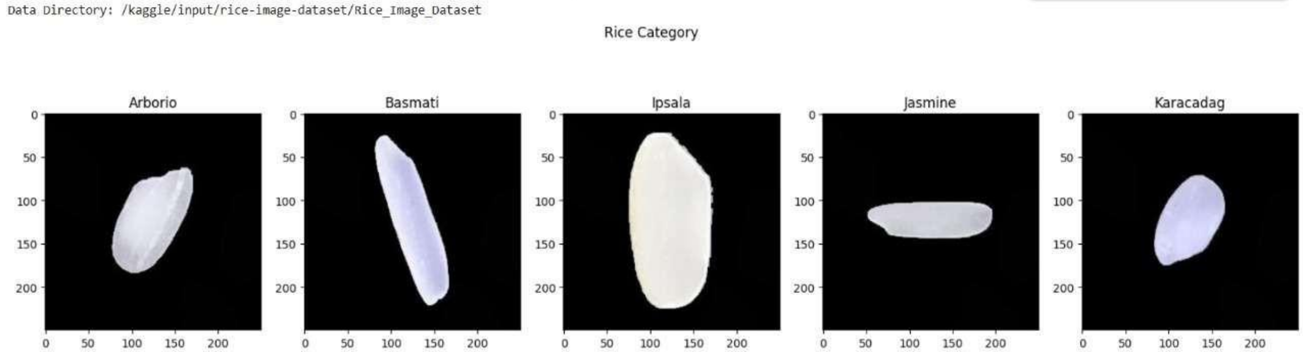
2	Tune the Model	• Hyperparameter Tuning: Used dropout rate, learning rate, number of layers, batch size • Validation Method: Used validation split (e.g., 0.2) and monitored validation accuracy during training with early stopping	 /usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these argumen Epoch 1/20 44/44 ━━━━━━━━━━ 84s 2s/step - accuracy: 0.7237 - loss: 0.7292 - val_accuracy: 0.9567 - val_loss: 0.1228 - learning_rate: 0.0010 Epoch 2/20 44/44 ━━━━━━━━━━ 75s 2s/step - accuracy: 0.9478 - loss: 0.1514 - val_accuracy: 0.9600 - val_loss: 0.0844 - learning_rate: 0.0010 Epoch 3/20 44/44 ━━━━━━━━━━ 75s 2s/step - accuracy: 0.9596 - loss: 0.1225 - val_accuracy: 0.9433 - val_loss: 0.1469 - learning_rate: 0.0010 Epoch 4/20 44/44 ━━━━━━━━━━ 83s 2s/step - accuracy: 0.9610 - loss: 0.1084 - val_accuracy: 0.9633 - val_loss: 0.0806 - learning_rate: 0.0010 Epoch 5/20 44/44 ━━━━━━━━━━ 76s 2s/step - accuracy: 0.9551 - loss: 0.1154 - val_accuracy: 0.9700 - val_loss: 0.0675 - learning_rate: 0.0010 Epoch 6/20 44/44 ━━━━━━━━━━ 75s 2s/step - accuracy: 0.9584 - loss: 0.1060 - val_accuracy: 0.9633 - val_loss: 0.0845 - learning_rate: 0.0010 Epoch 7/20 44/44 ━━━━━━━━━━ 82s 2s/step - accuracy: 0.9695 - loss: 0.1033 - val_accuracy: 0.9700 - val_loss: 0.0628 - learning_rate: 0.0010 Epoch 8/20 44/44 ━━━━━━━━━━ 76s 2s/step - accuracy: 0.9793 - loss: 0.0708 - val_accuracy: 0.9533 - val_loss: 0.0893 - learning_rate: 0.0010 Epoch 9/20 44/44 ━━━━━━━━━━ 75s 2s/step - accuracy: 0.9721 - loss: 0.0694 - val_accuracy: 0.9667 - val_loss: 0.0731 - learning_rate: 0.0010 Epoch 10/20 44/44 ━━━━━━━━━━ 76s 2s/step - accuracy: 0.9832 - loss: 0.0569 - val_accuracy: 0.9600 - val_loss: 0.0770 - learning_rate: 2.0000e-04

7. RESULTS

7.1 Output Screenshots

Deep Learning Code Outputs:-

```
Data Directory: /kaggle/input/rice-image-dataset/Rice_Image_Dataset
```



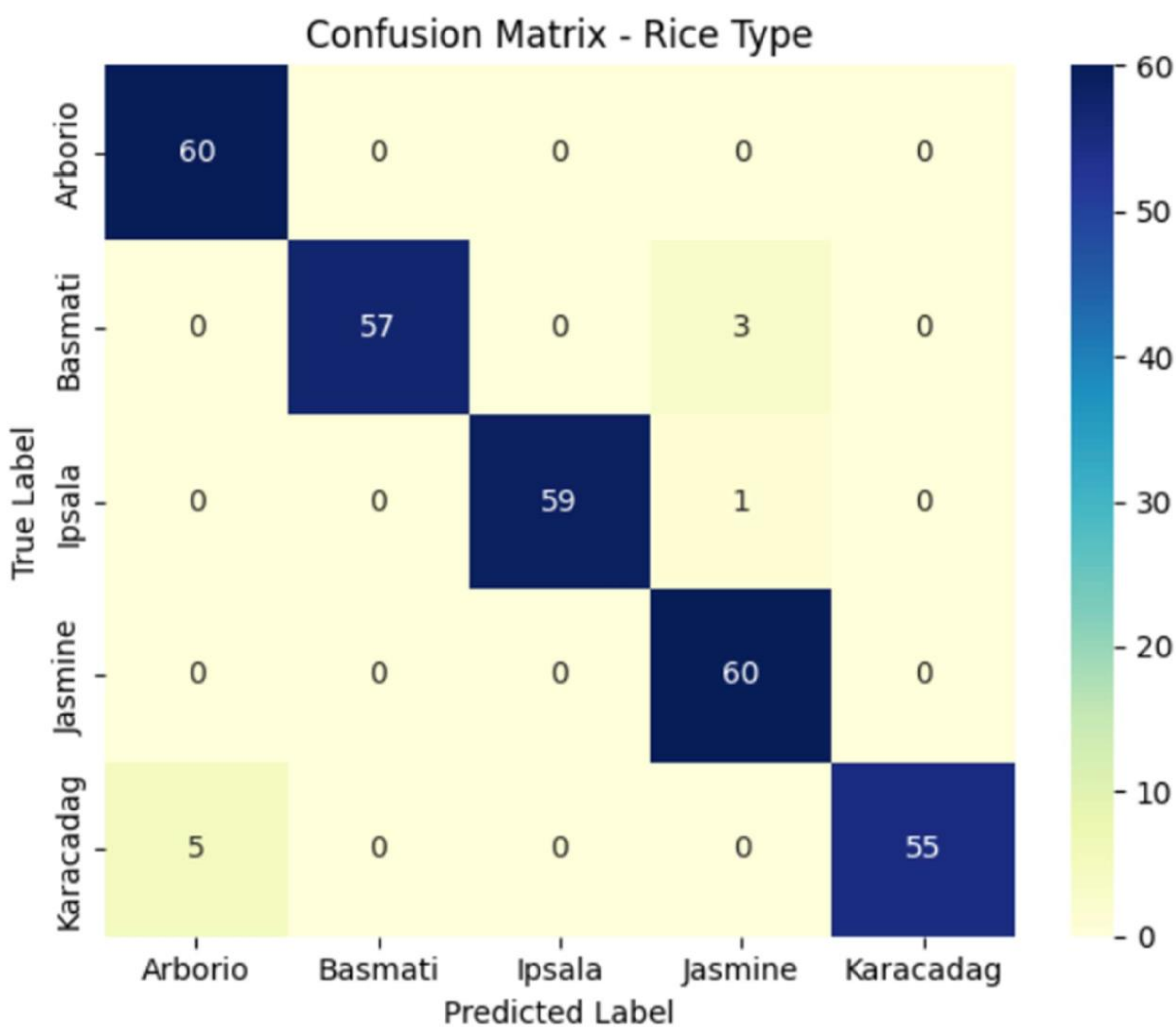

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning:

Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these argumer

Epoch 1/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 84s 2s/step - accuracy: 0.7237 - loss: 0.7292 - val_accuracy: 0.9567 - val_loss: 0.1228 - learning_rate: 0.0010
Epoch 2/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 75s 2s/step - accuracy: 0.9478 - loss: 0.1514 - val_accuracy: 0.9600 - val_loss: 0.0844 - learning_rate: 0.0010
Epoch 3/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 75s 2s/step - accuracy: 0.9596 - loss: 0.1225 - val_accuracy: 0.9433 - val_loss: 0.1469 - learning_rate: 0.0010
Epoch 4/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 83s 2s/step - accuracy: 0.9610 - loss: 0.1084 - val_accuracy: 0.9633 - val_loss: 0.0806 - learning_rate: 0.0010
Epoch 5/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 76s 2s/step - accuracy: 0.9551 - loss: 0.1154 - val_accuracy: 0.9700 - val_loss: 0.0675 - learning_rate: 0.0010
Epoch 6/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 75s 2s/step - accuracy: 0.9584 - loss: 0.1060 - val_accuracy: 0.9633 - val_loss: 0.0845 - learning_rate: 0.0010
Epoch 7/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 82s 2s/step - accuracy: 0.9695 - loss: 0.1033 - val_accuracy: 0.9700 - val_loss: 0.0628 - learning_rate: 0.0010
Epoch 8/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 76s 2s/step - accuracy: 0.9793 - loss: 0.0708 - val_accuracy: 0.9533 - val_loss: 0.0893 - learning_rate: 0.0010
Epoch 9/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 75s 2s/step - accuracy: 0.9721 - loss: 0.0694 - val_accuracy: 0.9667 - val_loss: 0.0731 - learning_rate: 0.0010
Epoch 10/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 76s 2s/step - accuracy: 0.9832 - loss: 0.0569 - val_accuracy: 0.9600 - val_loss: 0.0770 - learning_rate: 2.0000e-04
```

```
✅ Final model saved as rice_model.keras
```

```
📊 Evaluation on Test Set:
10/10 ━━━━━━━━━━━━━━━━━━━━ 13s 979ms/step - accuracy: 0.9699 - loss: 0.0690
Test Accuracy: 97.00%
```

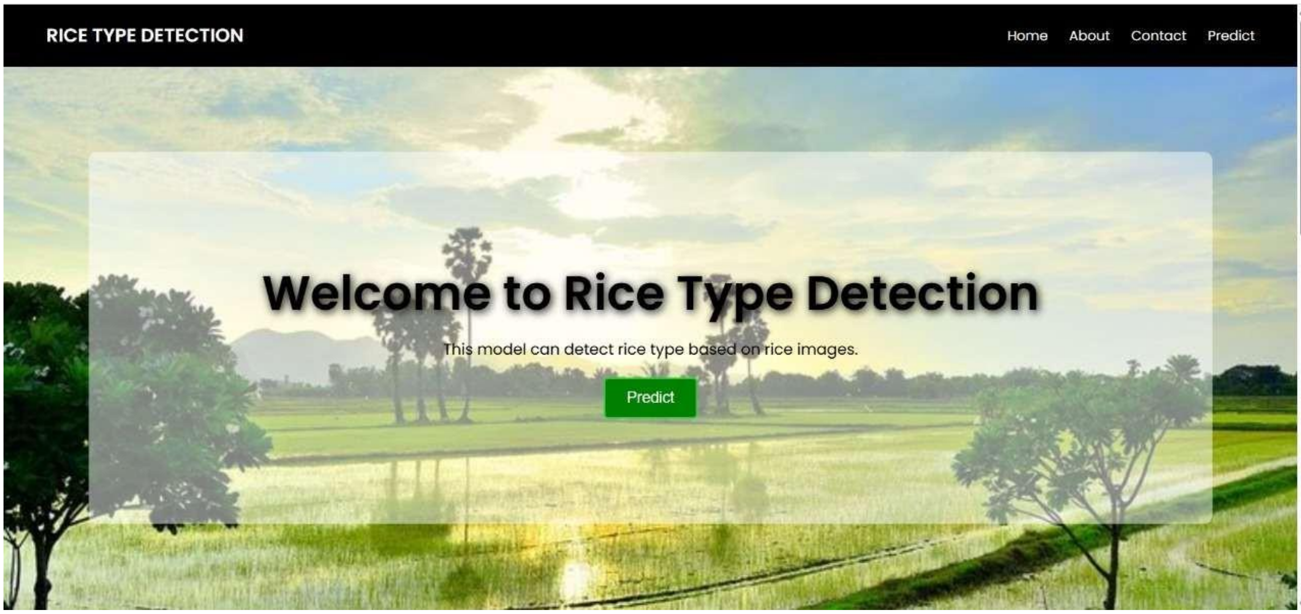
📋 Classification Report:

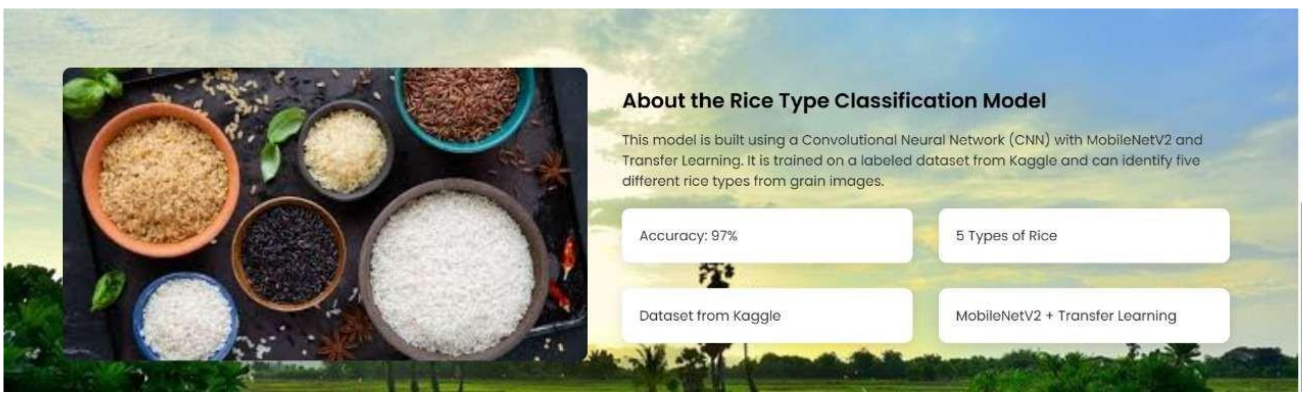
	precision	recall	f1-score	support
Arborio	0.92	1.00	0.96	60
Basmati	1.00	0.95	0.97	60
Ipsala	1.00	0.98	0.99	60
Jasmine	0.94	1.00	0.97	60
Karacadag	1.00	0.92	0.96	60
accuracy			0.97	300
macro avg	0.97	0.97	0.97	300
weighted avg	0.97	0.97	0.97	300

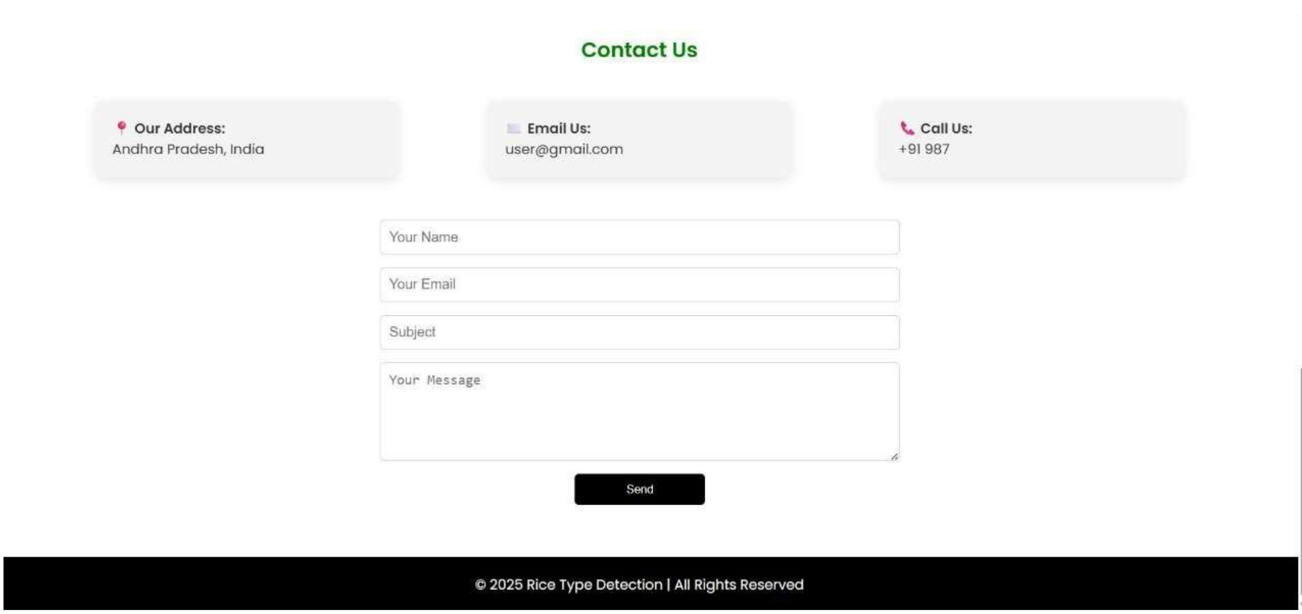


Website:-

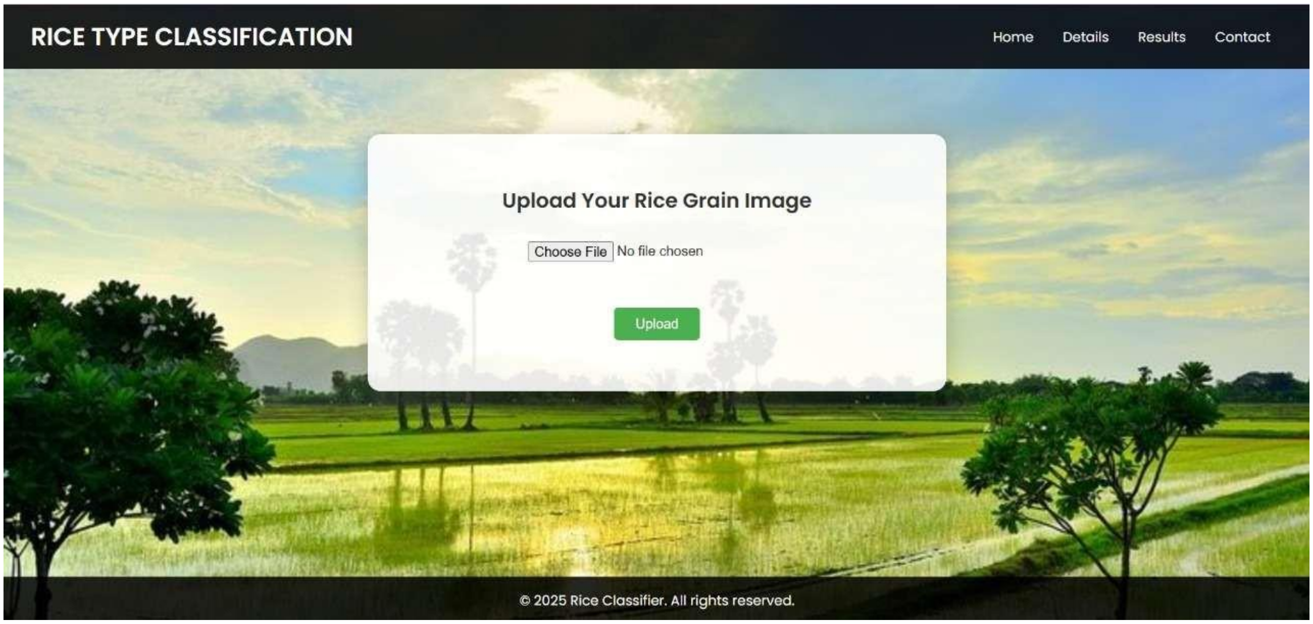
Index.html:-



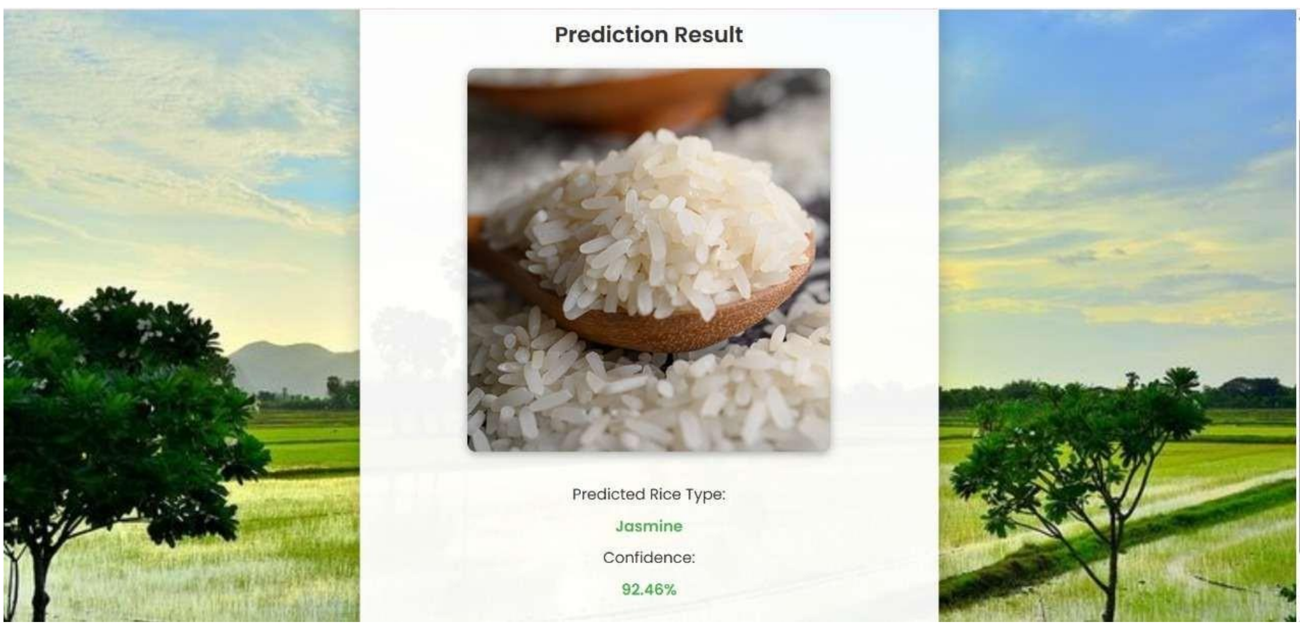




details.html:-



results.html:-



8. ADVANTAGES & DISADVANTAGES

8.1 ADVANTAGES:

1. Accurate Rice Type Identification

- Uses AI to correctly identify rice types from images.
- Reduces human errors in classification.

2. Saves Time and Effort

- Fast results within seconds.
- No need for manual checking or lab tests.

3. Better Market Pricing

- Helps farmers and traders know the exact variety.
- Allows fair and transparent pricing in the market.

4. Supports Better Crop Planning

- Recommends suitable rice types based on soil and climate.
- Helps farmers choose the best variety for their land.

5. Mobile-Friendly and Easy to Use

- Works on smartphones.
- Simple interface with image upload and voice/text options.

6. Multilingual Support

- Available in local languages for wider reach.
- Farmers can use it easily without needing English.

7. Offline and Low Network Usage

- Works in areas with poor or no internet.
- Stores data locally and syncs when online.

8. Supports Sustainability

- Suggests rice types that use less water or are climate-friendly.
- Promotes smart farming and resource conservation.

9. Helps Researchers and Officers

- Useful tool for agriculture officers and researchers.
- Can analyze regional trends and support planning.

10. Scalable and Future-Ready

- Can be expanded to support more grains (wheat, millet, etc.).
- Can include features like disease detection, yield prediction, etc.

DISADVANTAGES:

1. Limited Image Quality Dependence:

Poor or unclear images can reduce accuracy of identification.

2. Similar Varieties Confusion:

Visually similar rice types may get misclassified.

3. Internet Required for Some Features:

Full functionality may need a stable internet connection.

4. Needs Regular Model Updates:

ML model must be retrained to include new or hybrid varieties.

5. Regional Language Gaps:

May not support all local dialects or farming terms initially.

6. Technology Adoption Barriers:

Some farmers may find it hard to trust or use digital tools.

7. Limited Data Access:

Accurate soil and climate data may not be available in all regions.

8. Hardware Limitations:

Older smartphones may struggle with app performance or image processing.

9. Model Training Cost:

Collecting and abelling rice grain datasets requires time and money.

10. Misuse or Fake Inputs:

Users may upload wrong or misleading data, affecting results.

9. CONCLUSION

GrainPalette provides a smart, AI-powered solution for identifying rice grain types accurately and efficiently. By combining image processing with crop data analysis, it helps farmers, traders, and agricultural officers make informed decisions about rice cultivation and marketing. The tool saves time, improves accuracy, supports sustainable farming, and simplifies crop planning. While challenges like image quality, data limitations, and user adoption exist, the solution holds great potential to modernize and digitize rice farming across diverse regions. With further improvements and wider access, GrainPalette can significantly enhance the agricultural value chain.

10. FUTURESCOPE

The future scope of GrainPalette is broad and promising. The system can be expanded to support a wider range of rice varieties, including region-specific, hybrid, and rare types, improving its accuracy and usefulness. Beyond rice, the platform can evolve into a multi-grain classifier, identifying other important crops like wheat, millet, and barley. Advanced features such as pest and disease detection using image analysis can be integrated to support overall crop health monitoring. The inclusion of AI-driven yield prediction models will further assist farmers in planning and resource management. Real-time crop monitoring using drone or satellite data can enhance field-level insights. To increase accessibility, voice-based assistance in local languages can help reach non-literate users. GrainPalette could also be connected to local marketplaces, allowing farmers to sell produce based on grain quality. Integration with lab test results will boost classification accuracy, while linking recommendations with government subsidy schemes can help farmers make cost-effective decisions. With region-wise customization, the platform can eventually scale globally, benefiting rice growers in other countrienns as well.

11. APPENDIX

Source Code :

https://colab.research.google.com/drive/15VeXoOpBkijsab4wI-NExbjNjSW0QGve

Dataset Link :

https://www.kaggle.com/datasets/muratkokludataset/rice-image-dataset

GitHub Link :

https://github.com/bhuvaneswarialamkonda/GrainPalette--A-Deep-Learning-Odyssey-In-Rice-Type-Classification-Through-Transfer-Learning/tree/main

Project Demo :

https://drive.google.com/file/d/14-Z9zSndIXs9tfYmVNzAstkoHOe4bT74/view?usp=drivesdk