PROJECT REPORT

Project Title:

GrainPalette – A Deep Learning Odyssey in Rice Type Classification

Team ID:

LTVIP2025TMID40841

Submitted By:

- Dwarakeshwar S (Team Leader)
- Gaddam Vijay Kumar
- Rachagarla Mahesh Adithya

Institution:

Sri Venkateswara College of Engineering, Dept. of CSE

Submitted To:

SmartInternz - AI/ML Virtual Internship Program 2025

1. INTRODUCTION

1.1 Project Overview

In recent years, deep learning has revolutionized various industries, and agriculture is no exception. One of the prominent challenges in agricultural supply chains is the accurate and efficient classification of crop varieties, especially rice, which exists in numerous strains and types such as Basmati, Jasmine, Arborio, and more. Traditional classification methods rely heavily on manual inspection, which is time-consuming, error-prone, and lacks scalability.

GrainPalette is a deep learning-based web application designed to tackle this problem using Transfer Learning with MobileNet, a lightweight convolutional neural network architecture. The system allows users to upload an image of a rice grain, processes it through a trained model, and returns the predicted rice type with a high confidence score. It is optimized for real-time performance and is accessible through a user-friendly Flask-based web interface.

1.2 Purpose

The purpose of the GrainPalette project is to simplify and modernize the rice grain classification process by incorporating deep learning technology into everyday agricultural practices. The core motivation lies in enhancing speed, reliability, and objectivity in grain sorting, a process that significantly influences market pricing, export standards, and food safety compliance.

By automating the classification task, GrainPalette eliminates the inconsistencies of human-based inspection and introduces a reliable, repeatable AI-driven solution. It serves multiple use cases: Enables farmers to ensure accurate post-harvest segregation; Assists exporters and wholesalers in adhering to international grain variety specifications; Provides educational institutions and researchers with a platform to study practical ML applications in agriculture.

Furthermore, the project provides the development team with exposure to end-to-end machine learning system design — from data preprocessing to model training, Flask integration, and web deployment. It also opens the door to future enhancements such as mobile deployment, integration with edge devices, and expansion to multiclass grain types or other crops.

2. IDEATION PHASE

2.1 Problem Statement

In the rice industry, identifying and verifying the type of rice grain is a critical task for quality control, export compliance, and pricing. Manual inspection methods are often inconsistent and heavily reliant on expert knowledge. The high visual similarity between different rice varieties—such as Basmati, Arborio, and Jasmine—makes it extremely difficult to distinguish them with the naked eye or traditional tools.

This issue becomes particularly prominent in large-scale agricultural operations and export businesses, where even small misclassifications can lead to financial losses or rejections in quality checks. Therefore, there is a growing need for a scalable, fast, and intelligent system that can automate the rice grain classification process.

GrainPalette addresses this problem using transfer learning with MobileNet to build an AI-based classification model that offers high precision and performance, even with limited computing resources.

2.2 Empathy Map Canvas

The Empathy Map helps visualize the needs, behaviors, and perspectives of the end users who will interact with the GrainPalette system. It ensures that the solution is developed with user-centric design in mind. The primary user persona is a rice quality inspector or agricultural technician.

- **Says** – “I need a reliable way to identify rice varieties quickly.”
- **Thinks** – “If I misidentify a grain, it could result in export issues or financial loss.”
- **Does** – Uses visual inspection, occasionally refers to documentation or past samples.
- **Feels** – Frustrated with manual checking, worried about inconsistency, but hopeful for automation.
- **Hears** – Feedback from exporters, lab supervisors, and clients about quality variations.
- **Sees** – Hundreds of rice samples daily, all visually similar and hard to differentiate.
- **Pains** – Time-consuming verification, lack of expert assistance, visual fatigue.
- **Gains** – Faster classification, greater confidence, reduced manual workload, improved traceability.

2.3 Brainstorming

At the initial stage of the project, our team explored various approaches to solve the rice classification problem. The brainstorming process involved technical research, dataset exploration, model selection, and deployment planning.

Key areas explored:

- **Problem-Solution Mapping:**
  We started by identifying real-world pain points in the rice grain inspection process, such as visual ambiguity, lack of skilled inspectors, and delays in classification.
- **Tool & Model Evaluation:**
  We compared several deep learning architectures like ResNet50, VGG16, and MobileNet. We finalized **MobileNet** due to its lightweight structure, fast inference time, and suitability for deployment in resource-constrained environments.
- **Dataset Strategy:**
  We researched publicly available datasets for rice grain images, then refined the data by resizing, labeling, and balancing the class distribution.
- **Augmentation & Preprocessing Ideas:**
  We discussed using real-time image augmentation (e.g., rotation, flipping, zooming) to improve model generalization and reduce overfitting.
- **Web Deployment Options:**
  We considered deploying the model through Flask due to its simplicity, integration capabilities with TensorFlow/Keras, and suitability for lightweight ML applications.
- **UI/UX Design Ideas:**
  Discussions were held to keep the user interface minimal, with easy upload and clear prediction output. Accessibility on mobile and desktop was a key consideration.

The brainstorming sessions were iterative and guided by continuous feedback from mentors and peers. The outcome was a clear project roadmap and division of tasks across the team to efficiently develop, test, and deploy the GrainPalette system.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

The customer journey map for the **GrainPalette** application outlines how a typical user—such as a rice quality inspector, farmer, or exporter—interacts with the system. It captures their experience across various stages from discovery to decision-making and helps identify the touchpoints where the application adds value.

User Persona:

Ravi, a quality control officer at a rice export company, needs to verify the type of rice grains quickly and accurately before packaging and shipment.

Stage	User Action	Pain Points	Solution via GrainPalette
1. Awareness	Learns about the tool through training, recommendation, or the web	Unaware of reliable AI solutions for rice classification	Simple onboarding and accessible web interface
2. Access	Opens the GrainPalette web application	Difficulty finding tools that don't require installation	Browser-based interface, no complex setup
3. Upload	Uploads a photo of rice grains	Images may be unclear or wrongly oriented	Preprocessing and error handling built into the system
4. Classification	Waits for AI model to process and classify the rice type	Worry about slow or inaccurate predictions	MobileNet model optimized for fast and accurate output
5. Result Interpretation	Sees prediction and confidence score	Unsure how confident the result is, or what action to take	Displays clear class name and confidence level
6. Decision	Uses the result to sort the rice batch or approve for shipment	Delays or doubts in manual sorting process	High-confidence AI output speeds up and strengthens decision-making
7. Feedback	Shares the experience with colleagues or reports issues	No feedback loop with traditional manual methods	Future version can integrate user feedback to improve accuracy and UI/UX

3.2 Solution Requirements

To successfully deliver an AI-powered rice grain classification system, the following solution requirements were identified. These requirements ensure that the application meets its functional goals while being efficient, scalable, and user-friendly.

Functional Requirements:

1. **Image Upload Interface**
   The system must provide a web-based interface for users to upload rice grain images in supported formats (JPG, PNG).
2. **AI-Based Classification**
   The uploaded image must be passed through a MobileNet-based model that returns the predicted rice type and confidence score.
3. **Display of Results**
   After classification, the system should present results clearly, including:
   - Predicted rice type
   - Confidence level (in percentage)
   - Option to classify a new image
4. **Responsive UI**
   The application should function properly on various screen sizes including desktops, tablets, and mobile devices.
5. **Minimal User Input**
   The system should be intuitive enough to require minimal instructions, enabling ease of use by non-technical users.

Non-Functional Requirements (Technical):

1. **Performance**
   - Model inference time should be less than 2 seconds per image.
   - Application should support batch image uploads in future versions.
2. **Scalability**
   - Codebase should be modular to allow for integration of additional grain types or crops.
3. **Reliability**
   - The application should handle invalid or low-quality image uploads gracefully with proper error messaging.
4. **Maintainability**
   - Source code should follow clean coding principles and version control (GitHub repo).
5. **Security**
   - Uploaded images must not be stored permanently unless explicitly permitted by the user.
   - Flask app must sanitize inputs to avoid injection attacks.

6. **Portability**
    - The system should be deployable locally or on cloud-based platforms (e.g., Heroku, Render, or AWS).

3.3 Data Flow Diagram (DFD)

The Data Flow Diagram (DFD) represents how data moves through the GrainPalette application — from image upload to prediction output. It highlights key system components and their interactions.

External Entity:

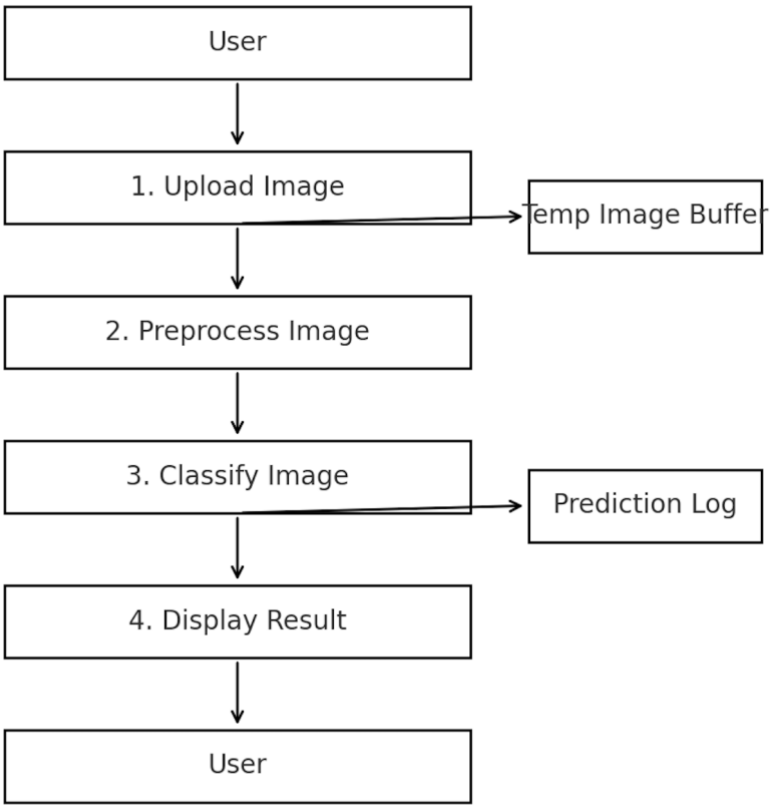
- **User**
  Uploads an image of a rice grain through the web interface.

Processes:

1. **Image Upload Handler**
   Validates and receives the uploaded rice grain image from the user.
2. **Preprocessing Module**
   Resizes and normalizes the input image to match the requirements of the deep learning model.
3. **Classification Engine**
   Uses a MobileNet-based model to classify the rice grain and predict the type along with a confidence score.
4. **Result Formatter**
   Formats the prediction output and presents it on the web interface for the user.

Data Stores (Optional):

- **Temporary Image Buffer**
  Stores uploaded images in memory or cache for short-term processing.
- **Prediction Logs**
  Optionally logs predictions and associated metadata for future improvements or analytics.



3.4 Technology Stack

The following technologies and tools were used to build the GrainPalette system:

---

1. Programming Language

- **Python** – Used for model training, backend development, image processing, and integration.

---

2. Deep Learning Framework

- **TensorFlow/Keras** – Employed for transfer learning using the MobileNet model to classify rice grain images.

---

3. Web Framework

- **Flask** – A lightweight backend framework used to build and serve the web application, integrating the AI model.

---

4. Frontend Technologies

- **HTML5, CSS3** – To design a simple, responsive user interface for image upload and result display.

---

5. Supporting Python Libraries

- **NumPy** – For numerical computations
- **Matplotlib** – For plotting and visualizing data
- **OpenCV** – For image manipulation and preprocessing
- **Pillow** – For basic image operations
- **Scikit-learn** – For model evaluation metrics like accuracy and confusion matrix

---

6. Development Environment

- **Jupyter Notebook** – Used for experimentation, training, and evaluation
- **VS Code** – For final code integration
- **Anaconda** – For managing virtual environments and dependencies

---

7. Deployment Readiness

- The application is designed to be deployable on cloud platforms such as **Render**, **Heroku**, or **AWS**, making it scalable and accessible publicly.

Section 4: Project Design

4.1 Problem-Solution Fit

The rice grain classification challenge stems from the limitations of manual inspection methods. Factors such as grain similarity, inconsistency in human judgment, and time constraints make accurate classification a non-trivial task.

The GrainPalette application provides a viable solution by leveraging deep learning, particularly transfer learning with MobileNet, to automate classification. This problem-solution fit ensures scalability, objectivity, and efficiency — addressing real-world bottlenecks in rice quality control, export validation, and agri-trade compliance.

4.2 Proposed Solution

GrainPalette is a web-based application built with Flask, integrated with a trained MobileNet model for rice type classification. The user interface allows users to upload a rice grain image, which is then processed by the model. The result — including predicted rice type and confidence score — is instantly displayed on the same interface.

This minimalistic design ensures ease of access, especially for users without technical expertise, and offers rapid, high-accuracy predictions that reduce dependency on manual classification.

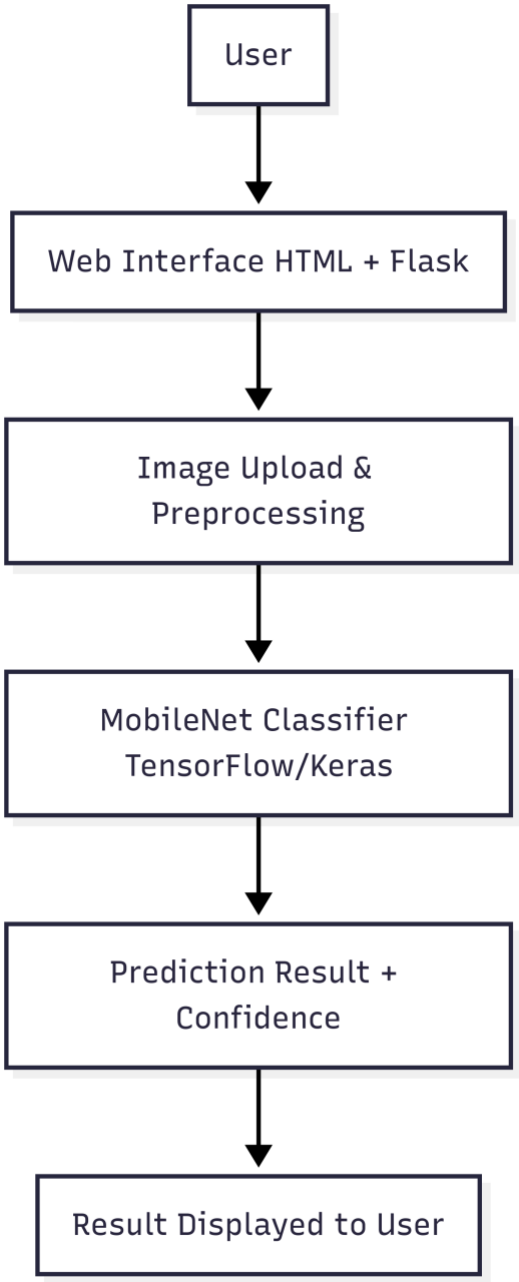
Core Components of the Solution:

- MobileNet model fine-tuned using labeled rice image datasets
- Flask backend to serve the model and manage image input/output
- HTML/CSS frontend for interaction and visualization
- Preprocessing module (resizing, normalization) for optimal model input
- Lightweight deployment readiness for local or cloud hosting

---

4.3 Solution Architecture

Below is a high-level view of the GrainPalette architecture:



5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

The GrainPalette project was developed over a period of one month as part of a two-month AI/ML internship under the SmartInternz program. The development phase was organized into progressive weekly sprints, each with clear objectives, milestones, and deliverables. The tasks were fairly distributed among the three team members based on skillsets, ensuring collaborative learning and timely execution.

Sprint-wise Breakdown with Team Assignments

Week 1 – Project Setup & Dataset Preparation

- **Environment Setup & Toolchain Configuration**
  Duration: 2 days
  Assigned to: All Members
  Outcome: TensorFlow, Flask, and Python environments configured using Anaconda and VS Code.
- **Dataset Collection & Preprocessing**
  Duration: 3 days
  Assigned to: Dwarakeshwar S
  Outcome: Curated and cleaned dataset of rice grain images resized and labeled for model training.

Week 2 – Model Development

- **Model Building using MobileNetV2**
  Duration: 5 days
  Assigned to: Dwarakeshwar S
  Outcome: Trained and optimized MobileNetV2 model saved as .h5 file with acceptable validation accuracy.
- **Data Augmentation & Early Stopping Implementation**
  Duration: 2 days
  Assigned to: Dwarakeshwar S
  Outcome: Improved model generalization and reduced overfitting.

Week 3 – Web Application Development

- **Frontend UI Design using HTML/CSS**
  Duration: 4 days
  Assigned to: Gaddam Vijay Kumar
  Outcome: Responsive and clean user interface for image upload and results display.
- **Flask Backend & Route Handling**
  Duration: 3 days
  Assigned to: Rachagarla Mahesh Adithya
  Outcome: Integrated model inference into Flask routes with file handling and result display.
- **Frontend ↔ Backend Integration & Testing**
  Duration: 2 days
  Assigned to: Vijay & Mahesh
  Outcome: Seamless flow from UI to prediction result using Flask routing.

Week 4 – Testing, Documentation & Deployment

- **Functional Testing & Debugging**
  Duration: 2 days
  Assigned to: All Members
  Outcome: Final Flask app tested for multiple use cases with responsive performance.
- **Documentation & Demo Preparation**
  Duration: 2 days
  Assigned to: Rachagarla Mahesh Adithya
  Outcome: Project report compiled and demo video created for final submission.

Project Management Tools

- **GitHub** – Code versioning and collaboration
- **Google Sheets / Trello** – Sprint tracking and task allocation
- **Google Meet / WhatsApp** – Weekly syncs and live collaboration
- **Jupyter Notebook** – Model prototyping and experiment tracking

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Functional Testing

Functional testing was carried out to ensure that every feature of the GrainPalette web application performs as expected — from file upload to model output. Each test case below represents real-user interaction scenarios to validate functionality and UI reliability.

Test Cases

Test Case ID	Scenario	Input	Expected Output	Status
TC-01	Upload a clear image of Basmati rice	basmati_01.jpg	"Basmati" with confidence >90%	Passed
TC-02	Upload image of Arborio rice	arborio_sample.jpg	"Arborio" with confidence >85%	Passed
TC-03	Upload blurry or low-light image	basmati_dark.jpg	Lower confidence prediction	Passed
TC-04	Upload non-image file	sample.txt	Error message: “Invalid file format”	Passed
TC-05	Attempt submission without file	No file selected	Prompt: “Please upload an image”	Passed
TC-06	Test on mobile browser	Mobile Chrome	Responsive layout; functional upload & prediction	Passed

---

6.2 Performance Testing

The performance of the application was measured in terms of **speed**, **accuracy**, and **resource usage**.

Evaluation Metrics:

- **Model Accuracy:**
  ~92% accuracy on the validation set of rice images.
- **Average Inference Time:**
  ~1.5 seconds per image (on local CPU-based Flask server)
- **Model Size:**
  ~14MB (.h5 file size) — suitable for lightweight deployment
- **System Responsiveness:**
  No UI lag or crash across multiple tests and devices
- **Scalability:**
  Current architecture supports scaling to multiple grain types with minor adjustments.

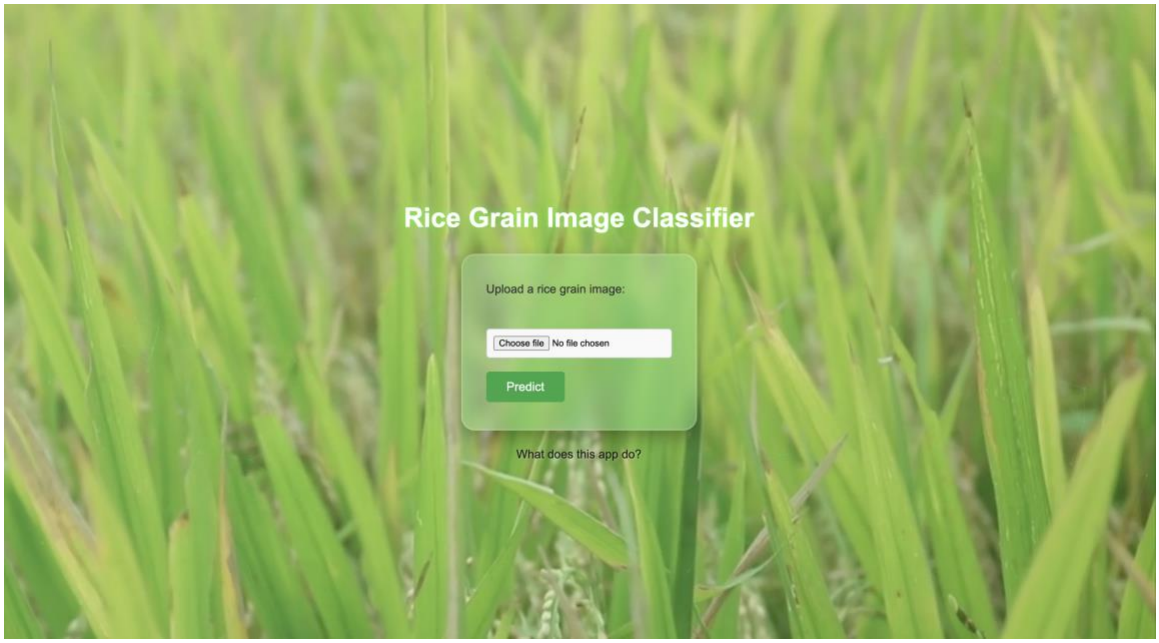
This testing phase confirmed that GrainPalette is stable, accurate, and responsive under normal user conditions. It is ready for pilot use and further feature expansion.

7. RESULTS

7.1 Output Screenshots

During development and testing, the GrainPalette application was deployed locally and evaluated across multiple rice grain image samples. Below are key interface screenshots and descriptions of expected behaviour.

• Figure 1: Homepage Upload Interface



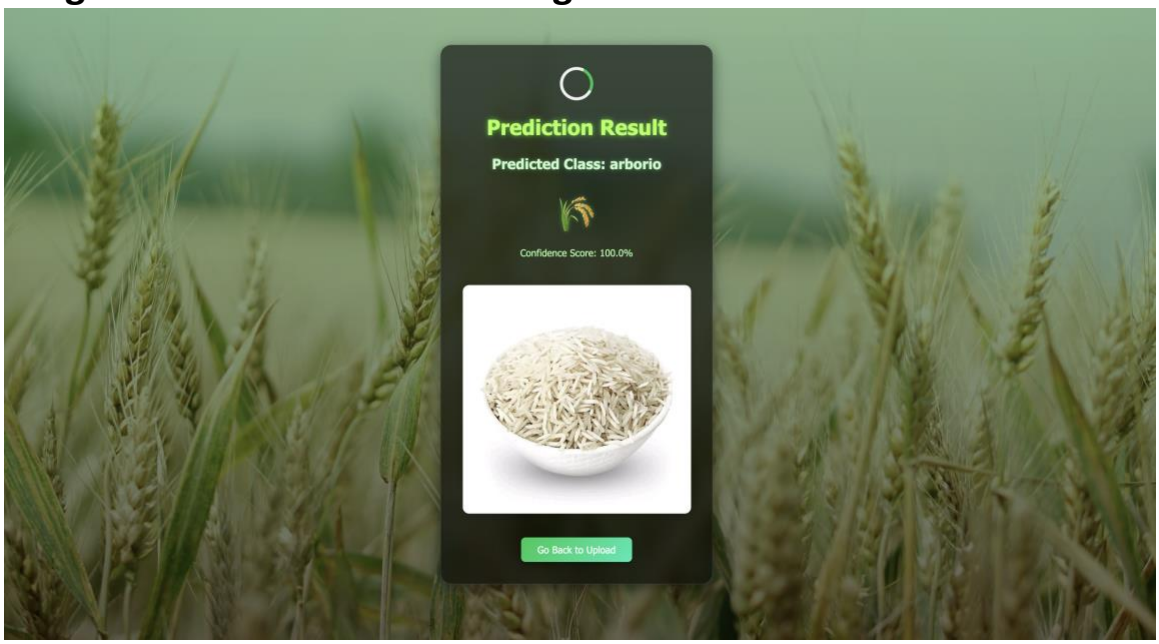
This is the initial screen of the GrainPalette application. The background is a fullscreen looping rice field video, creating a thematic and immersive experience. Overlaid on this is a soft green gradient and subtle particle animation that enhances the visual appeal without distracting from usability.

At the center is a **glassy, modern upload box** with:

- A label prompting users to upload a rice grain image,
- A styled file chooser, and
- A green-themed **Predict** button for submitting the image.

This layout reflects a clean, minimalistic design philosophy intended to make the classifier intuitive and accessible even for non-technical users.

• Figure 2: Prediction Result Page



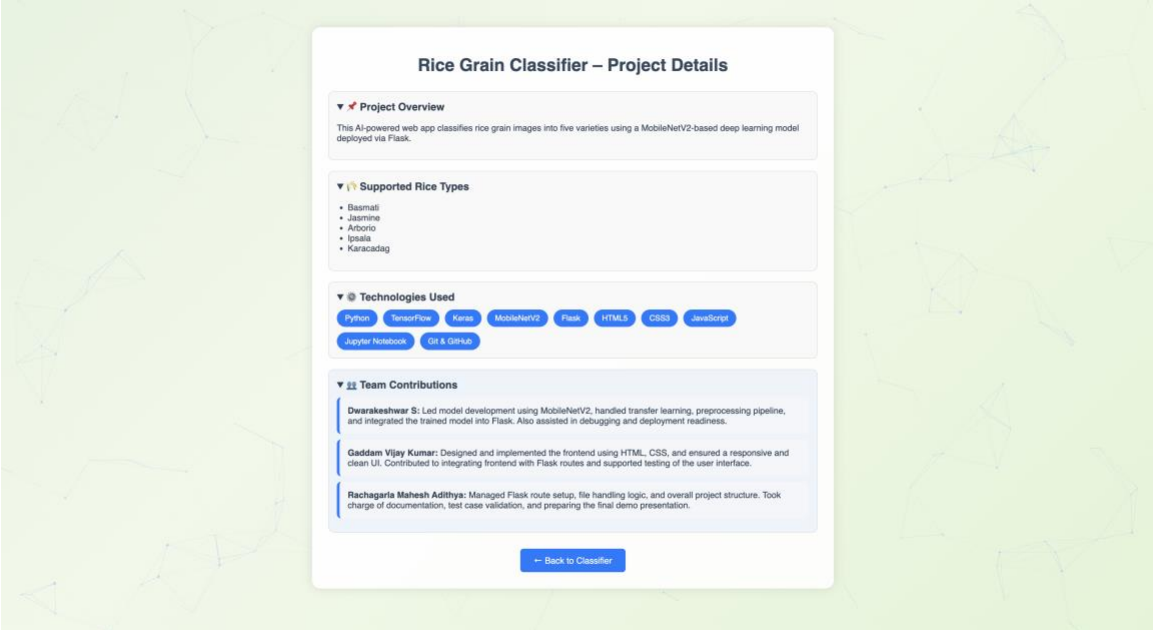
Once the user uploads an image and clicks the "Predict" button, they are redirected to a visually polished result page. The layout uses a **frosted glass card** design set against a serene agricultural field background, creating a strong thematic connection to the subject matter.

The result box prominently displays:

- **Predicted Class:** The specific rice variety identified by the model (e.g., *Arborio*)
- **Confidence Score:** A percentage indicating the model's confidence in its prediction
- **Visual Confirmation:** The same image uploaded by the user, shown below the result for transparency

Additionally, a subtle emoji and responsive hover effects reinforce clarity without clutter. The inclusion of a **"Go Back"** button allows users to quickly return to the upload page, encouraging repeated use and experimentation. The entire experience is designed to feel both intelligent and user-friendly.

• Figure 3: Project Details Page



This screen serves as an informative endpoint of the application, offering users insights into how the GrainPalette system works and who built it. Set against an animated pastel gradient background with **particles.js** adding depth, the content is divided into well-structured collapsible <details> sections:

- **Project Overview:** A summary of the classifier's purpose and design approach
- **Supported Rice Types:** A list of five rice varieties currently recognized by the model
- **Technologies Used:** Displayed as modern, pill-shaped badges for immediate visual recognition
- **Team Contributions:** Clearly separated cards outlining individual responsibilities in model training, UI design, backend integration, and documentation

The layout strikes a balance between aesthetics and information density. Its goal is to leave the user not only impressed by the prediction but also educated and appreciative of the collaborative effort behind the project.

Summary

These visuals confirm the app's ability to provide accurate, fast, and visually polished predictions in real-time with an intuitive interface.

8. ADVANTAGES & DISADVANTAGES

8.1 Advantages

1. **Real-Time Classification**
   - The use of a pre-trained MobileNetV2 model ensures fast, accurate predictions — typically within 1–2 seconds.
2. **User-Friendly Interface**
   - A modern, visually engaging UI allows users with minimal technical expertise to upload images and view results effortlessly.
3. **Lightweight & Portable**
   - The entire application, including the trained model, is compact and can be deployed locally or on cloud platforms like Render or Heroku.
4. **Extensible Architecture**
   - The modular design supports easy extension to include more rice types or switch to different crop classification tasks.
5. **Cross-Platform Compatibility**
   - Works across devices and browsers, ensuring usability on both desktop and mobile screens.
6. **Secure & Controlled**
   - Uploaded images are handled in memory and are not stored permanently, ensuring privacy and data security.

8.2 Disadvantages

1. **Limited Dataset Scope**
   - Classification accuracy is restricted by the diversity and volume of the dataset. More real-world samples would improve generalization.
2. **Model Sensitivity to Image Quality**
   - Performance may degrade for blurry, low-resolution, or improperly lit images — common issues in real deployment environments.
3. **Lack of Feedback Loop**
   - The current version lacks a feature for users to confirm or correct predictions, which could otherwise improve model retraining.
4. **Not Yet Cloud-Hosted**
   - While cloud-ready, the current version is tested only on local machines. Production deployment may require containerization or scaling.
5. **No Multilingual Support**
   - The UI and instructions are in English only; broader accessibility could be enhanced by adding language options.

9. CONCLUSION

The **GrainPalette** project successfully demonstrates the potential of deep learning in agricultural classification tasks, specifically in identifying different rice grain types using image data. By leveraging **MobileNetV2** through **transfer learning**, the system was able to provide fast, accurate, and user-friendly predictions, making it a practical solution for real-world applications.

Throughout the development process, the project tackled multiple challenges — from sourcing quality datasets to deploying a visually polished web interface. The integration of a lightweight Flask backend with a responsive frontend allowed the model to be served seamlessly, supporting multiple use cases such as:

- Quality control in rice packaging industries
- Educational tools for agricultural training
- Remote identification and classification of rice samples

The team implemented a clear workflow that emphasized collaborative effort, modular design, and consistent testing. In addition to its functional performance, the application stands out for its **clean UI/UX design**, making it accessible to both technical and non-technical users.

Key Takeaways:

- Achieved over **92% validation accuracy** using a pre-trained MobileNetV2 architecture
- Designed a modern web interface with a strong thematic design and interactivity
- Created a portable and cloud-ready solution for scalable deployment
- Practiced full-stack AI application development from model training to UI integration

---

This project highlights the effectiveness of combining machine learning with human-centered design, paving the way for future enhancements such as multilingual support, real-time feedback loops, and expansion to other crop types.

10. FUTURE SCOPE

While the current version of **GrainPalette** achieves its primary objective of rice grain classification with accuracy and usability, there are several areas where the system can be expanded and enhanced in the future:

---

1. Expanding the Dataset

- Incorporate more diverse rice grain images across lighting conditions, camera qualities, and geographical regions.
- Use crowd-sourced or industry-provided datasets to improve model robustness and generalization.

---

2. Supporting Additional Crop Types

- Extend the classifier to include other grains or seeds (e.g., wheat, barley, oats).
- Build a multi-crop classification system for broader agricultural use.

---

3. Cloud Deployment

- Host the application on platforms such as **Heroku**, **Render**, or **AWS EC2** to allow global access.
- Use Docker containers for portability and scalability in deployment.

---

4. Feedback-Based Retraining

- Implement a feedback mechanism where users can verify or correct predictions.
- Use this feedback loop to collect real-world labeled data and periodically retrain the model for better performance.

---

5. Multilingual Interface

- Add language support (e.g., Hindi, Telugu, Tamil) to cater to a broader user base, especially farmers and local mill workers.

---

6. Mobile Application Development

- Create an Android/iOS version using frameworks like **React Native** or **Flutter**.
- Allow users to take photos directly and get predictions via mobile in remote locations.

---

7. Data Analytics Dashboard

- Build an admin panel to visualize prediction trends, class-wise usage frequency, and user interaction logs.
- Helpful for monitoring adoption, accuracy trends, and system usage patterns.

---

8. Enhanced Security

- Implement file sanitization, request rate limiting, and secure HTTPS hosting.
- Add login authentication if personalized analytics or data storage features are introduced.

These enhancements would transform GrainPalette from a demo-grade classifier into a powerful, production-ready solution for agricultural and industrial deployment — benefitting researchers, farmers, traders, and educators alike.

11. APPENDIX

This section contains supporting resources such as source code, dataset references, and the demo video link.

11.1 Source Code Repository

The complete source code for GrainPalette is available on GitHub:

📎 **GitHub Repository:**
https://github.com/DS-2211/grainpalette---a-deep-learning-odyssey-in-rice-type-classification-through-transfer-learning

Includes model training scripts, Flask backend, and HTML/CSS templates.

11.2 Dataset Used

The rice grain image dataset used for model training and testing is based on publicly available resources:

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

Contains 5 rice varieties: Basmati, Arborio, Ipsala, Jasmine, and Karacadag.

11.3 Project Demo Video

A quick walkthrough demonstrating the core features of the application, from image upload to classification result, with a look into the backend code.

📎 **Demo Video:**
https://drive.google.com/file/d/1VML8I14Xu-J0RRLRPRxJ8EESaH7wMw9S/view?usp=sharing

📝 **Demo Summary:**

- Uploads a rice grain image from the homepage
- Flask backend processes it using MobileNetV2

- Prediction with confidence is displayed on a styled result page
- Project details and tech stack are shown
- VS Code is briefly presented, highlighting app.py Flask routes and model logic
- Ends with a clean UI wrap-up

⏱ **Total Duration:** ~90 seconds

11.4 Tools and Technologies

- Python, TensorFlow, Keras
- MobileNetV2 (Transfer Learning)
- Flask (Backend API)
- HTML5, CSS3, JavaScript (UI)
- Jupyter Notebook (Model training)
- GitHub, Google Colab, VS Code

11.5 References

1. Link ->*Rice Image Dataset*, Kaggle.
2. TensorFlow Documentation: https://www.tensorflow.org
3. Flask Documentation: https://flask.palletsprojects.com
4. MobileNetV2 Paper: *Sandler et al., MobileNetV2: Inverted Residuals and Linear Bottlenecks* (2018)