Project Design Phase

Solution Architecture

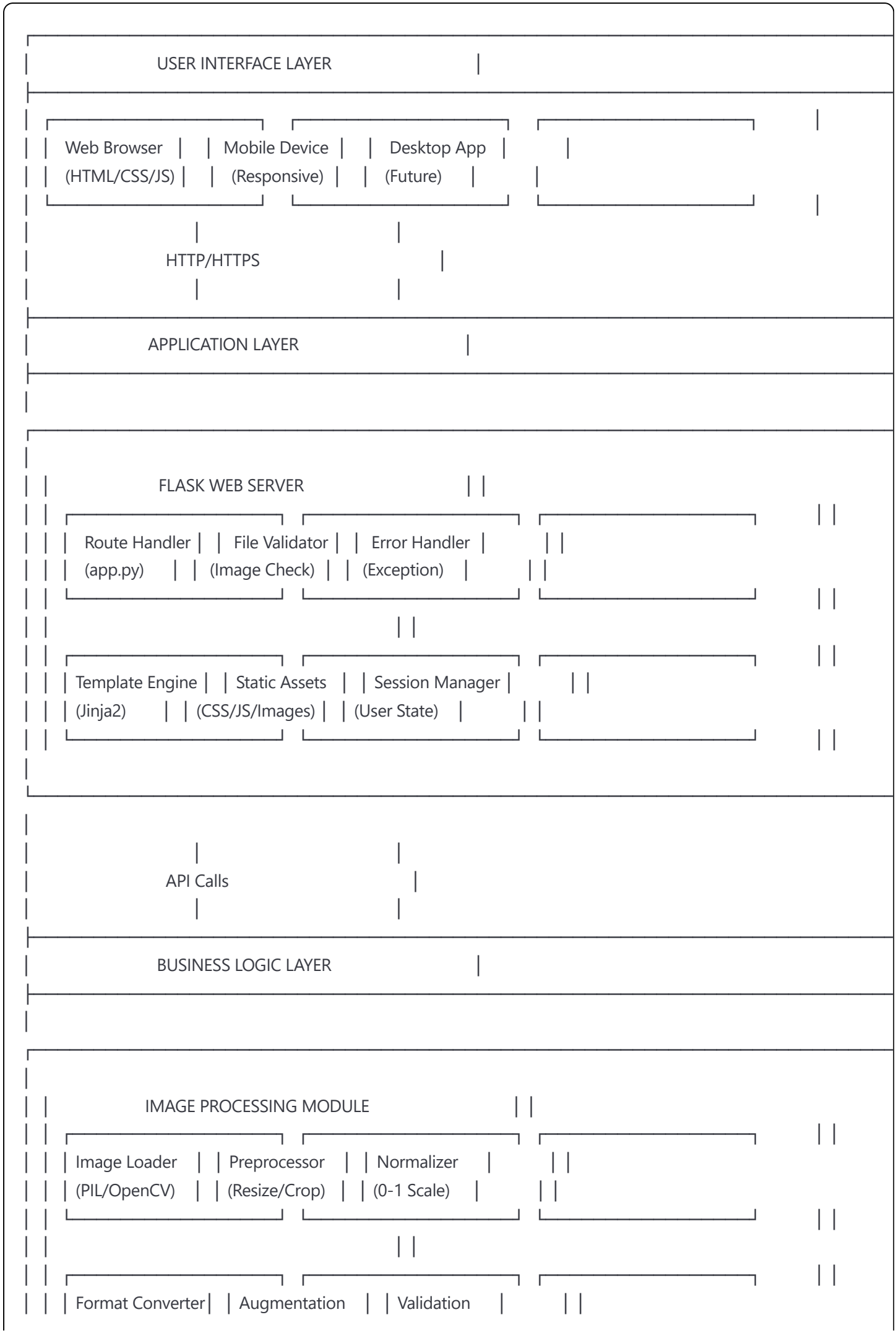
Date: 15 February 2025
Team ID: LTVIP2025TMID32946
Project Name: GrainPalette - A Deep Learning Odyssey In Rice Type Classification Through Transfer Learning
Maximum Marks: 4 Marks

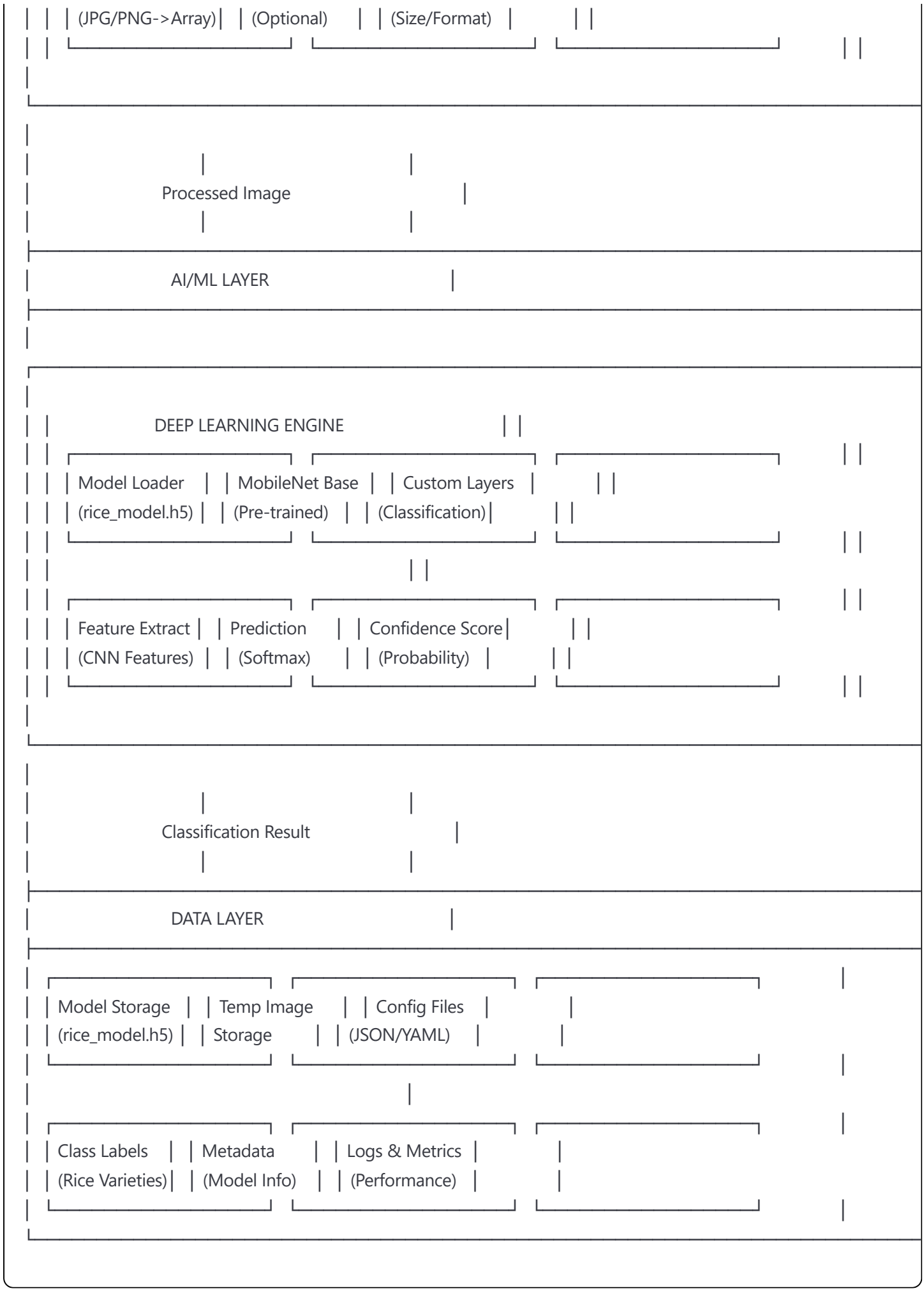
Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

GrainPalette Solution Architecture Diagram:


USER INTERFACE LAYER
Web Browser
(HTML/CSS/JS)
Mobile Device
(Responsive)
Desktop App
(Future)
HTTP/HTTPS
APPLICATION LAYER
FLASK WEB SERVER
Route Handler
(app.py)
File Validator
(Image Check)
Error Handler
(Exception)
Template Engine
(Jinja2)
Static Assets
(CSS/JS/Images)
Session Manager
(User State)
API Calls
BUSINESS LOGIC LAYER
IMAGE PROCESSING MODULE
Image Loader
(PIL/OpenCV)
Preprocessor
(Resize/Crop)
Normalizer
(0-1 Scale)
Format Converter
Augmentation
Validation



Architecture Components Description:

1. User Interface Layer

- **Web Browser Interface**: HTML5/CSS3 responsive design
- **Mobile Support**: Optimized for smartphones and tablets
- **Interactive Elements**: File upload, drag-and-drop, result display

2. Application Layer (Flask Framework)

- **Route Management**: Handles GET/POST requests for home and prediction
- **File Validation**: Checks image format, size, and validity
- **Template Rendering**: Jinja2 for dynamic HTML generation
- **Static Asset Management**: CSS, JavaScript, and image files
- **Error Handling**: Graceful error management and user feedback

3. Business Logic Layer

- **Image Processing Pipeline**:
  - Load and validate uploaded images
  - Resize to 224x224 pixels (MobileNet input size)
  - Normalize pixel values to 0-1 range
  - Convert to appropriate tensor format
- **Data Validation**: Format checking and preprocessing validation

4. AI/ML Layer (Deep Learning Engine)

- **Model Architecture**: MobileNet-based CNN with transfer learning
- **Feature Extraction**: Convolutional layers for rice grain pattern recognition
- **Classification Head**: Custom dense layers for rice variety prediction
- **Inference Engine**: TensorFlow/Keras model execution
- **Confidence Scoring**: Softmax probabilities for prediction certainty

5. Data Layer

- **Model Storage**: Persistent storage for trained model (rice_model.h5)
- **Temporary Storage**: Uploaded images (auto-cleanup)
- **Configuration**: Model parameters and class labels
- **Logging**: Performance metrics and error tracking

Data Flow Process:

1. **Image Upload**: User uploads rice grain image through web interface

2. **Validation**: Flask validates file format, size, and content
3. **Preprocessing**: Image resized, normalized, and formatted for model input
4. **Model Inference**: Deep learning model processes image and generates predictions
5. **Post-processing**: Extract top prediction with confidence score
6. **Result Display**: Flask renders result page with prediction and original image
7. **Cleanup**: Temporary files removed after processing

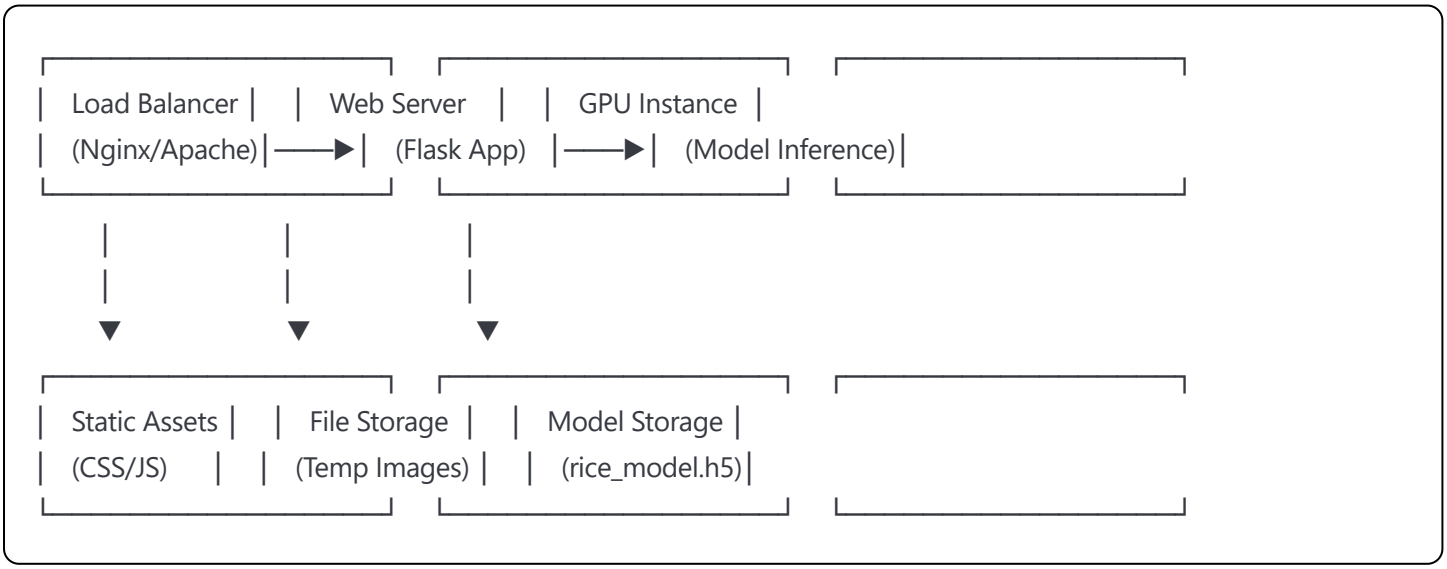
Technology Stack Integration:

Layer	Technology	Purpose
Frontend	HTML5, CSS3, JavaScript	User interface and interaction
Backend	Flask (Python)	Web server and API handling
AI/ML	TensorFlow, Keras	Deep learning model inference
Image Processing	PIL, NumPy	Image manipulation and preprocessing
Model	MobileNet + Custom CNN	Rice classification algorithm

Performance Considerations:

- **Scalability**: Stateless design for horizontal scaling
- **Efficiency**: MobileNet for fast inference (<3 seconds)
- **Memory Management**: Automatic cleanup of temporary files
- **Error Resilience**: Comprehensive error handling and logging
- **Security**: Input validation and file type restrictions

Deployment Architecture:



Security & Quality Assurance:

- **Input Validation**: File type, size, and content validation
- **Error Handling**: Graceful degradation and user feedback
- **Model Security**: Secure model storage and access control
- **Data Privacy**: No persistent storage of user images
- **Performance Monitoring**: Logging and metrics collection