# Requirement Analysis – Technology Stack

Project Name: GrainPalette – A Deep Learning Odyssey in Rice Type Classification through Transfer Learning

Team ID: LTVIP2025TMID32428

Date: 31 January 2025

## Technical Architecture

The GrainPalette system is designed using a client-server architecture integrating a deep learning model for rice grain type classification. The frontend interacts with the backend via REST APIs, and the backend handles AI inference, authentication, and database operations. The application is deployed on cloud infrastructure to ensure scalability and availability.

[Insert Architecture Diagram Here]

## Table-1: Components & Technologies

|  |  |  |
| --- | --- | --- |
| S.No | Component Description | Technology |
| 1 | User Interface - Web UI for uploading rice images, viewing predictions, and history. | HTML, CSS, JavaScript, React.js |
| 2 | Application Logic-1 - Handles image upload, preprocessing, and API requests. | Node.js, Express.js |
| 3 | Application Logic-2 - AI inference service to classify rice type. | Python, TensorFlow/Keras |
| 4 | Database - Stores user data, prediction results, and image metadata. | MongoDB (NoSQL) |
| 5 | Cloud Database - Managed database service for scalability. | MongoDB Atlas |
| 6 | File Storage - Stores uploaded images securely. | AWS S3 / Local Storage |
| 7 | External API-1 - Image preprocessing or enhancement APIs (optional). | OpenCV API, Pillow |
| 8 | Machine Learning Model - Classifies rice type using transfer learning. | ResNet50 / EfficientNet (Keras/TensorFlow) |
| 9 | Infrastructure (Server / Cloud) - Deployment environment. | AWS EC2 / Azure VM / Local Server |

## Table-2: Application Characteristics

|  |  |  |
| --- | --- | --- |
| S.No | Characteristics Description | Technology |
| 1 | Open-Source Frameworks - Frameworks and libraries used. | React.js, Node.js, Express.js, TensorFlow, Keras |
| 2 | Security Implementations - Authentication, data encryption, and API security. | JWT Authentication, bcrypt hashing, HTTPS, OWASP practices |
| 3 | Scalable Architecture - Cloud-ready design ensuring horizontal scalability. | Microservices architecture, Docker, Kubernetes |
| 4 | Availability - Ensures minimal downtime through redundancy. | AWS Load Balancer, Multi-zone deployment |
| 5 | Performance - Optimized for fast inference and minimal latency. | Model optimization, CDN for static assets, Redis caching |