**Project Design Phase-II**
**Technology Stack (Architecture & Stack)**

| | |
|---|---|
| Date | 27 June 2025 |
| Team ID | LTVIP2025TMID43995 |
| Project Name | Transfer Learning-based Classification of Poultry Diseases for Enhanced Health Management |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

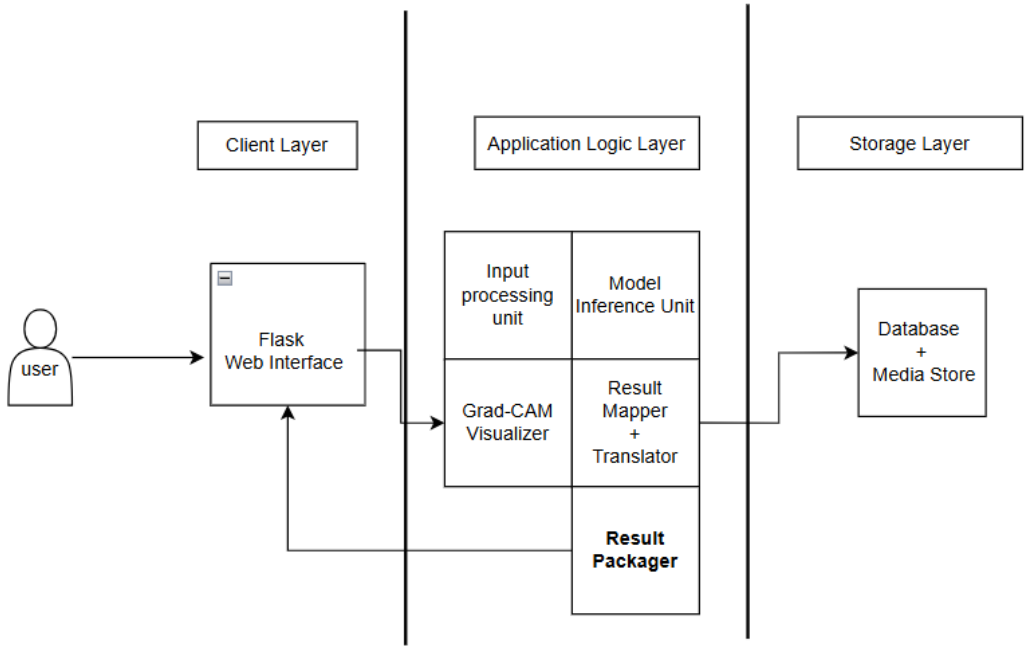

**Table-1 : Components & Technologies:**

| S.No | Component | Description | Technology |
|---|---|---|---|
| 1. | User Interface | Web interface for farmers and vets to upload images and input symptoms | HTML, CSS, Bootstrap, JavaScript |
| 2. | Application Logic-1 | Backend logic for routing, form handling, and inference calling | Python with Flask Framework |
| 3. | Application Logic-2 | Disease prediction logic using transfer learning | TensorFlow / Keras (ResNet50 / EfficientNetB0) |
| 4. | Application Logic-3 | Explainable AI output using Grad-CAM | OpenCV, Matplotlib |
| 5. | Database | Store user info, logs, history, and feedback | SQLite (Local) or MongoDB |
| 6. | File Storage | Storage of uploaded images and Grad-CAM outputs | Local File System / AWS S3 / Firebase Storage |
| 7. | External API-1 | For language translation and multilingual support | Google Translate API |
| 8. | Machine Learning Model | Used for poultry disease classification into 4 categories | Pre-trained CNN + Fine-tuned ResNet50 Model |

**Table-2: Application Characteristics:**

| S.No | Characteristics | Description | Technology |
|---|---|---|---|
| 1. | Open-Source Frameworks | Web, AI, and Visualization frameworks used | Flask, TensorFlow, Keras, Bootstrap |
| 2. | Security Implementations | Login/Authentication, HTTPS, input validation, secure data handling | Flask Login, SHA-256, HTTPS (TLS) |
| 3. | Scalable Architecture | 3-tier structure: UI → Application Server → Model & DB | Flask App with modular MVC separation |
| 4. | Availability | Can be deployed on cloud or offline; availability depends on hosting medium | Heroku / AWS / Firebase with fallback |
| 5. | Performance | Optimized ML model, image compression, caching for Grad-CAM outputs, fast REST calls | Model quantization, local cache, Flask |