**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 27 June 2025 |
| Team ID | LTVIP2025TMID34533 |
| Project Name | Pollen Profiling |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

**User Interface:**

* Web-based UI
* Flask-rendered templates

**Application Logic:**

* **React**: For REST API, web app integration
* **Python (TensorFlow / PyTorch)**: For pollen image classification model
* **Image Preprocessing (OpenCV / PIL)**: Resize, normalize, augment

**Database & Storage:**

* **Local DB**: SQLite for prototype / MySQL for production
* **Cloud DB (optional)**: AWS RDS / Firebase (if scaling needed)
* **File Storage**: Local filesystem or AWS S3 (for image storage)

**Machine Learning Model:**

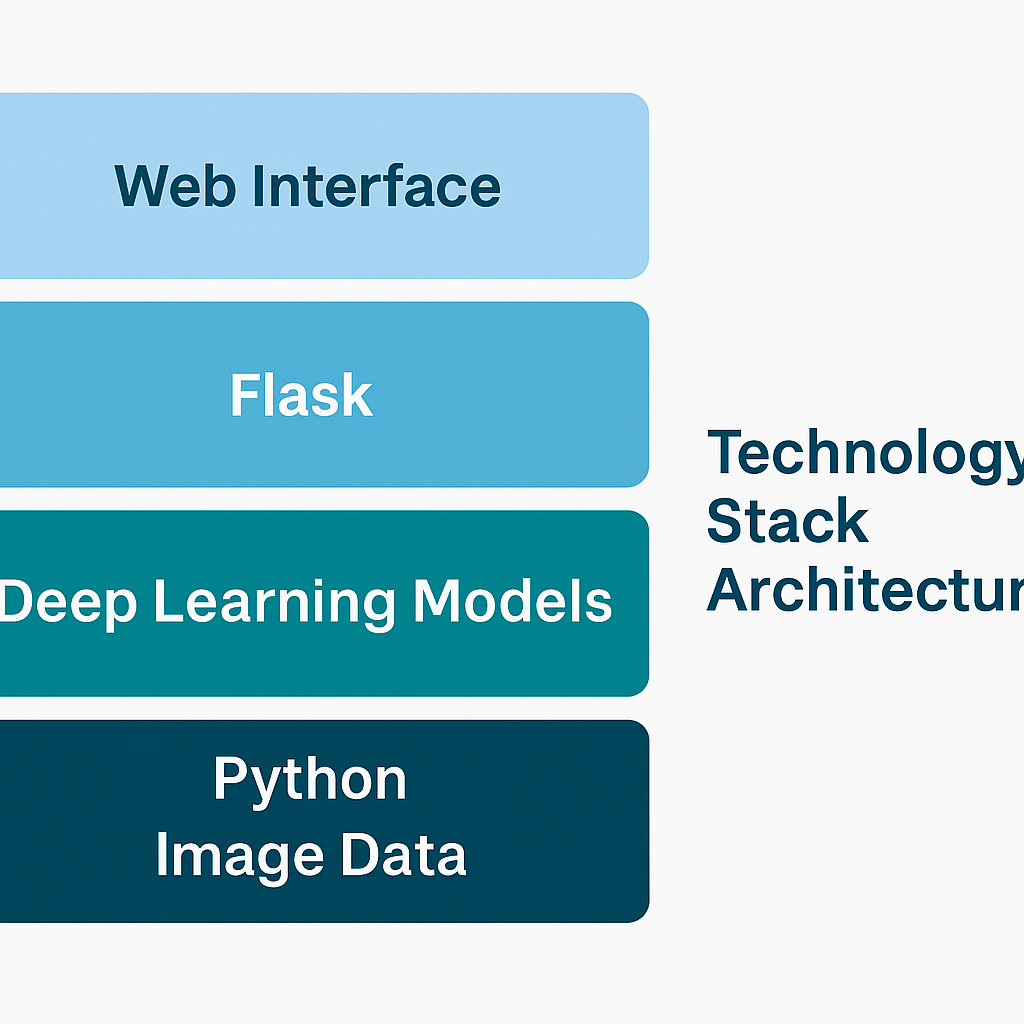
* CNN-based model (TensorFlow / PyTorch) deployed as a service
* Exposed via Flask API

**Infrastructure:**

* Deployment on local server / AWS EC2 / Heroku for cloud hosting
* Scalable using Docker, Kubernetes (optional for large-scale deployment)

**External Interfaces (if any):**

* Email service (Gmail API / SMTP for registration confirmation)
* (Optional) Google Maps API for pollen distribution visualization



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

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | Web-based UI for interaction | HTML, CSS, JavaScript, Bootstrap |
|  | Application Logic-1 | API and server logic | React |
|  | Application Logic-2 | Image preprocessing & ML integration | Python (OpenCV, TensorFlow / PyTorch) |
|  | Database | User info, logs | SQLite / MySQL |
|  | File Storage | Pollen image storage | Local FS / AWS S3 |
|  | Machine Learning Model | Classify pollen images | TensorFlow / PyTorch CNN model |
|  | Infrastructure | Local / cloud deployment | Local server / AWS EC2 / Heroku |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | Frameworks used | Flask, TensorFlow, PyTorch, OpenCV |
|  | Security Implementations | Data protection measures | HTTPS, JWT tokens, SHA-256 password hashing, IAM (if on cloud) |
|  | Scalable Architecture | Supports growing data & users | 3-tier architecture, Docker (optional microservices) |
|  | Availability | Continuous access | Cloud hosting (AWS/Heroku) + optional load balancers |
|  | Performance | Designed for speed | Caching (Flask-Caching), image preprocessing optimizations |