Project Design Phase-II

# Technology Stack (Architecture & Stack)

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Team ID | Project Name | Maximum Marks |
| 10 June 2025 | LTVIP2025TMID33677 | HematoVision | 4 Marks |

Technical Architecture:

Example: Blood Cell Classification System using Transfer Learning

Reference: HematoVision.ipynb

Guidelines:

* Include all processes (Image Input, Pre-processing, Model Inference, Result Display)
* Provide infrastructural demarcation (Local / Cloud)
* Indicate external interfaces (ML Model, APIs)
* Indicate data storage and model serving
* Indicate interface to ML model for inference

## Table-1: Technology Components

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Component | Description | Technology Used |
| 1 | User Interface | Interface for user to upload microscopic images and view results | HTML, CSS, JavaScript, React.js |
| 2 | Application Logic-1 | Image Upload and Preprocessing | Python, OpenCV |
| 3 | Application Logic-2 | Model Inference using Transfer Learning | TensorFlow / PyTorch |
| 4 | Application Logic-3 | Display of classified cell type and confidence score | Flask / Django |
| 5 | Database | To store metadata, prediction logs | SQLite / Firebase Realtime DB |
| 6 | Cloud Database | Cloud-based database for storing results and large image datasets | Firebase / Google Cloud Firestore |
| 7 | File Storage | Storage of uploaded blood smear images | Google Cloud Storage / Firebase Storage |
| 8 | External API-1 | External model hosting or prediction APIs (if used) | TensorFlow Serving / HuggingFace APIs |
| 9 | Machine Learning Model | Deep learning model for WBC classification (CNN + Transfer Learning) | EfficientNet / ResNet (Keras / PyTorch) |
| 10 | Infrastructure (Cloud) | Cloud setup for deployment and model serving | Google Colab / Heroku / Render / AWS |

## Table-2: Application Characteristics

|  |  |  |
| --- | --- | --- |
| S.No | Characteristics | Description & Technology |
| 1 | Open-Source Frameworks | TensorFlow, Keras, Flask, OpenCV - Python Ecosystem |
| 2 | Security Implementations | Limited access via API keys, HTTPS - SSL, Firebase Auth |
| 3 | Scalable Architecture | Modular architecture - REST API structure |
| 4 | Availability | Hosted on cloud with 99.9% uptime - Firebase Hosting / AWS Lambda |
| 5 | Performance | Model quantization, lazy loading, image resizing - CDN, Flask cache |

## References:

* https://keras.io/guides/transfer\_learning/
* https://colab.research.google.com/
* https://firebase.google.com/docs
* https://www.tensorflow.org/tfx
* https://huggingface.co/models