**HEMATOVISION: Advanced Blood Cell Classification Using Transfer Learning**

**Team ID**: LTVIP2025TMID42255  
**Date**: July 17, 2025

# **Data Flow Diagrams & Technical Architecture**

## **Page 1: Context-Level Data Flow Diagram (DFD - Level 0)**

The Context-Level DFD represents the entire system as a single process and shows the interaction between the system and external entities.

### **External Entities**:

* **User**: Inputs blood smear image
* **Medical Database/API**: Provides labeled data and diagnostic history

### **Process**:

* **Blood Cell Classification System**

### **Data Stores**:

* **Image Repository**
* **ML Model Output Storage**

### **Data Flows**:

* User → Uploads Image → System
* System → Processes Image → Sends to ML Model
* ML Model → Classifies Image → Sends result to UI & Storage
* System → Fetches Data → Medical Database/API

## **Page 2: Level-1 Data Flow Diagram**

This diagram elaborates the internal processes of the system.

### **Processes**:

1. **Image Upload Module**
2. **Preprocessing Engine**
3. **Transfer Learning Model Inference**
4. **Result Visualizer and Report Generator**

### **Data Stores**:

* **Raw Image DB**
* **Processed Image DB**
* **Result Logs**

### **Data Flows**:

* User uploads image → Image Upload Module → Raw Image DB
* Raw image → Preprocessing Engine → Processed Image DB
* Processed image → ML Inference → Result Logs
* Results → Visualizer → User interface
* User request history → Result Logs
* External APIs → Medical DB Query → ML Model

## **Page 3: Technical Architecture with Tables**

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

| S.No | Component | Description | Technology |
| --- | --- | --- | --- |
| 1 | User Interface | Front-end for uploading images, viewing results | HTML, CSS, JavaScript, React.js |
| 2 | Application Logic-1 | Image preprocessing and normalization | Python (OpenCV, NumPy) |
| 3 | Application Logic-2 | Model loading and inference using transfer learning | TensorFlow, PyTorch |
| 4 | Database | Storing metadata and classification results | MongoDB / NoSQL |
| 5 | File Storage | Store uploaded blood smear images | IBM Cloud Object Storage |
| 6 | External API-1 | Fetching diagnostic guidelines | WHO/CDC API |
| 7 | Machine Learning Model | Blood cell classification | Pre-trained CNN (e.g., ResNet50) |
| 8 | Infrastructure | Deployment infrastructure | IBM Cloud / Kubernetes |

### **Table-2: Application Characteristics**

| S.No | Characteristics | Description | Technology |
| --- | --- | --- | --- |
| 1 | Open-Source Frameworks | Frontend and ML frameworks | React.js, TensorFlow, Flask |
| 2 | Security Implementations | Access controls, API security, and encryption | JWT, OAuth 2.0, SSL/TLS |
| 3 | Scalable Architecture | Modular, microservices-oriented with containerization | Docker, Kubernetes |
| 4 | Availability | Cloud-based deployment with redundancy | Load Balancers, Multi-Zone Nodes |
| 5 | Performance | Efficient image processing and caching for faster inference | Redis Cache, CDN, Model Optim. |

**End of Document**