

# Solution Architecture: HematoVision - Advanced Blood Cell Classification

Date	10 June 2025
Team ID	LTVIP2025TMID33915
Project Name	Hematovision: Advanced Blood Cell Classification
Maximum Marks	4 marks

## 1. Introduction

This document outlines the solution architecture for HematoVision, an AI-powered system designed for the accurate and efficient classification of blood cells. The architecture leverages a combination of deep learning models and a user-friendly web application to provide a robust and scalable solution for pathologists and healthcare professionals.

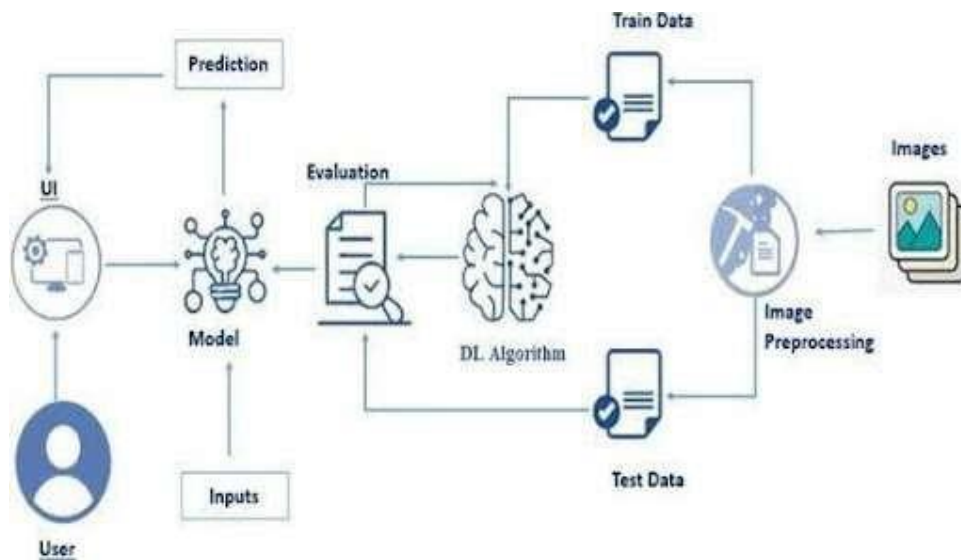
## 2. High-Level Architecture Overview

The HematoVision system follows a typical client-server architecture, where a web-based frontend interacts with a Python-based backend that hosts the machine learning model.

The core components include:

- **Client-Side (Web Browser):** User interface for interacting with the system.
- **Web Application Backend (Flask):** Handles user requests, manages image uploads, and orchestrates interactions with the machine learning model.

- **Machine Learning Model (TensorFlow/Keras):** The trained deep learning model responsible for blood cell classification.
- **Storage:** For temporary storage of uploaded images.



### 3. Detailed Component Breakdown

#### 3.1. Client-Side (Frontend)

- **Technology:** HTML, CSS, JavaScript (standard web technologies).
- **Purpose:** Provides the graphical user interface (GUI) for users to interact with the HematoVision system.
- **Key Functions:**
  - **Image Upload:** Allows users to select and upload blood cell images (e.g., home.html ).
  - **Display Results:** Presents the classification prediction and the uploaded image (e.g., result.html ).

- **User Feedback:** Potentially provides visual cues for upload progress or errors.

### 3.2. Web Application Backend (Flask)

- **Technology:** Python, Flask framework ( `app.py` ).
- **Purpose:** Acts as the central hub, receiving requests from the frontend, processing them, and returning responses. It integrates the machine learning model.
- **Key Functions:**
- **API Endpoints:** Defines routes for image upload ( `/predict` ) and serving web pages ( `/` ).
- **Image Handling:** Receives uploaded image files, saves them temporarily, and prepares them for model inference.
- **Model Inference Orchestration:** Loads the pre-trained `blood_cell.h5` model and passes the processed image data to it for classification.
- **Result Processing:** Receives the prediction from the model and formats it for display on the frontend.
- **Error Handling:** Manages invalid file types or other processing errors.
- **Templating:** Renders HTML templates ( `home.html` , `result.html` ) to serve dynamic content to the user.

### 3.3. Machine Learning Model

- **Technology:** TensorFlow, Keras, MobileNetV2 ( `model.ipynb` , `blood_cell.h5` ).
- **Purpose:** The core intelligence of the system, responsible for accurately classifying blood cells.

- **Key Functions:**
- **Image Classification:** Takes a preprocessed blood cell image as input and outputs a probability distribution over the four blood cell classes (Eosinophil, Lymphocyte, Monocyte, Neutrophil).
- **Feature Extraction:** The pre-trained MobileNetV2 acts as a powerful feature extractor.
- **Prediction:** Provides the final predicted class based on the model's output.
- **Training Details (from model.ipynb and README\_HematoVision.md ):**
- **Architecture:** MobileNetV2 (pre-trained) with custom classification layers.
- **Dataset:** 12,500 augmented blood cell images from Kaggle.
- **Training:** 5 epochs (as per README), Adam optimizer, categorical cross-entropy loss.
- **Accuracy:** Approximately 85.3% validation accuracy.
- **Model Persistence:** The trained model is saved as blood\_cell.h5 for deployment.

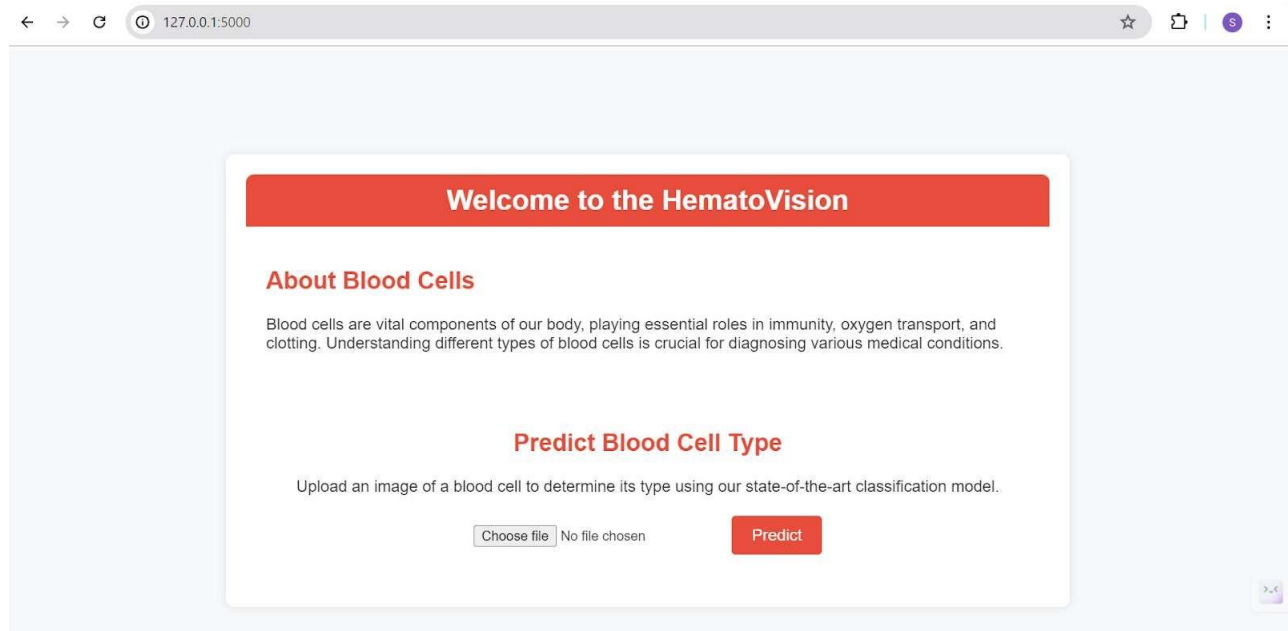
### 3.4. Storage

- **Technology:** Local filesystem.
- **Purpose:** Temporarily stores uploaded images before they are processed by the model.
- **Key Functions:**
- **Temporary Uploads:** The static/uploads/ directory is used to store images uploaded by users via the web interface.

- **File Management:** Images are typically deleted after processing or after a certain period to manage storage space.

## 4. Data Flow and Interactions

1. **User Interaction:** A user accesses the HematoVision web application through their browser, which loads `home.html` from the Flask backend.
2. **Image Upload:** The user selects a blood cell image and uploads it via the web form. This HTTP POST request is sent to the `/predict` endpoint of the Flask application.
3. **Backend Processing:**
  - The Flask application ( `app.py` ) receives the uploaded image.
  - It saves the image temporarily in the `static/uploads/` directory.
  - The image is then preprocessed (resized, normalized) to match the input requirements of the MobileNetV2 model.
  - The preprocessed image is passed to the loaded `blood_cell.h5` model for inference.
4. **Model Prediction:** The MobileNetV2 model performs the classification and returns the predicted blood cell type.
5. **Result Display:**
  - The Flask application receives the prediction from the model.
  - It then renders the `result.html` template, passing the prediction and the path to the uploaded image.
  - The `result.html` page is sent back to the user's browser, displaying the classification outcome.



## 5. Deployment Considerations

- **Containerization:** The application can be containerized using Docker for consistent deployment across different environments.
- **Cloud Platforms:** Suitable for deployment on cloud platforms like Render, Railway, or Heroku, as indicated in README\_HematoVision.md .
- **Scalability:** For high traffic, the Flask application can be scaled horizontally, and the model inference can be offloaded to dedicated GPU-enabled services.
- **Security:** Implement secure coding practices, input validation, and secure handling of uploaded files.

## 6. Future Enhancements

- **API for Integration:** Develop a dedicated API for external systems to integrate with the classification service.

- **Batch Processing:** Allow for the upload and classification of multiple images simultaneously.
- **Confidence Scores:** Display confidence levels for predictions to aid pathologists in decision-making.
- **Database Integration:** Store classification results and metadata in a database for auditing and analysis.
- **User Authentication:** Implement user login and role-based access control for secure access.