🩸 HAMEOTOVISION

ADVANCED BLOOD CELL CLASSIFICATION USING TRANSFER LEARNING

https://github.com/GoddetiPavithra/Haemotovision

📘 PROJECT REPORT

This document presents HematoVision, an advanced Artificial Intelligence and Deep Learning based web application designed to classify human blood cell types automatically using Transfer Learning techniques.
The system utilizes a pre-trained Convolutional Neural Network model (such as VGG16 / MobileNetV2) to accurately identify and classify microscopic blood cell images. The primary objective of this project is to assist medical professionals, pathologists, and diagnostic laboratories by improving efficiency, reducing manual effort, and enhancing diagnostic accuracy in hematology.
HematoVision addresses critical challenges in traditional blood cell analysis, where manual observation is time-consuming and prone to human error. By leveraging modern machine learning algorithms, the system provides fast, reliable, and automated predictions through a user-friendly web interface.
Developed as part of the Bachelor of Computer Applications (BCA) curriculum, this project demonstrates the practical application of Artificial Intelligence in the healthcare domain.

Project Details

HematoVision: Advanced Blood Cell Classification Using Deep Deep Learning

Developed By: Goddeti Pavithra

Course: Bachelor of Computer Applications (BCA)

Hall Ticket No: 23420161047009

Institution: Sri Shankaranda Giri Swamy Degree College, Guntakal

Academic Year: 2025–2026

Team Information:

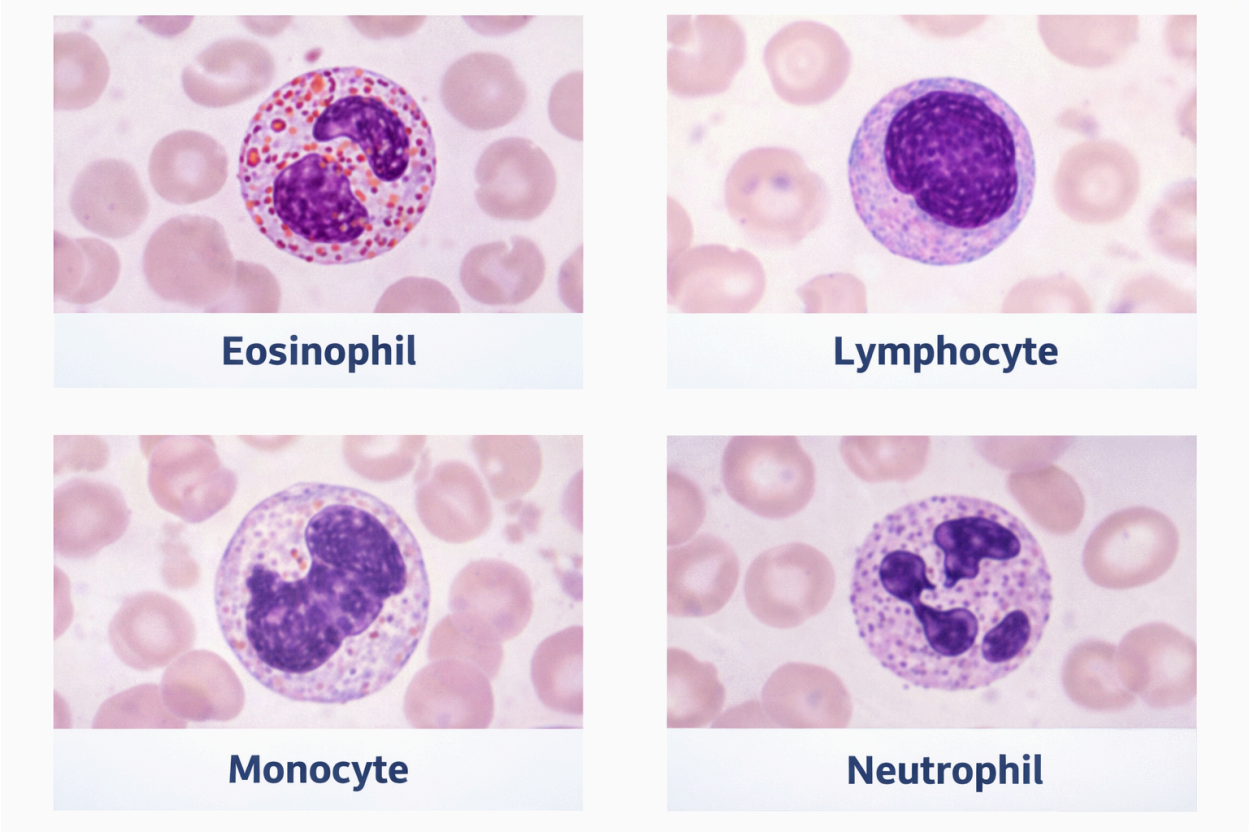
Team ID: LTVIP2026TMIDS49741

Team Size: 4 Members

1. INTRODUCTION

1.1 Project Overview

Haemotovision is an advanced blood cell classification system developed using Transfer Learning and Deep Learning techniques. The system is designed to automatically classify different types of blood cells from microscopic images using Convolutional Neural Networks (CNN).
Manual classification of blood cells is time-consuming and requires expert knowledge. This project aims to automate the process using pre-trained deep learning models to improve accuracy and reduce diagnosis time.
The model is trained on a labeled dataset of blood cell images and can classify the following types:



- Eosinophil
- Lymphocyte
- Monocyte
- Neutrophil

1.2 Purpose

The main purpose of this project is:

To build an intelligent system for automatic blood cell classification.
To reduce human error in laboratory analysis.
To assist medical professionals in faster diagnosis.
To apply transfer learning for improved performance with limited dataset.

2. IDEATION PHASE

2.1 Problem Statement

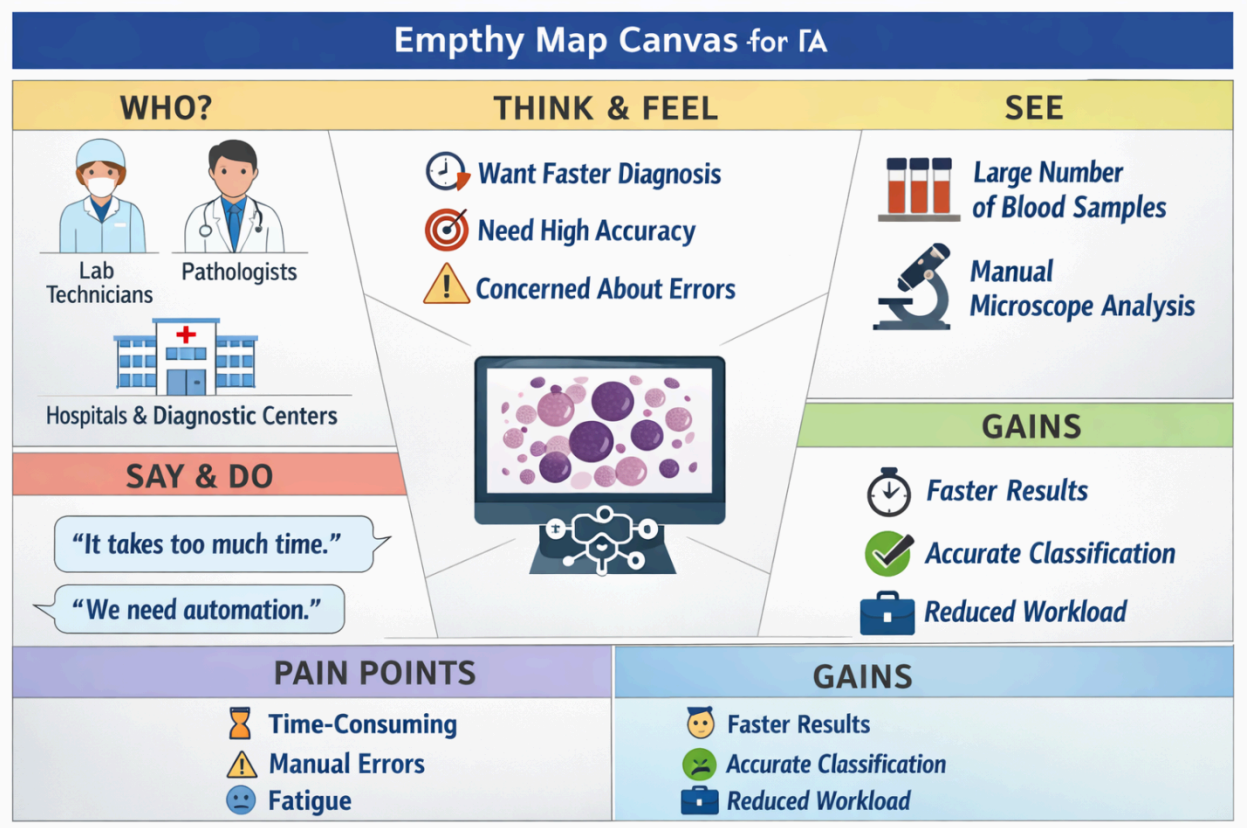
Blood cell classification plays a critical role in medical diagnosis and disease detection. Traditionally, this process is performed manually by trained pathologists using microscopic examination. However, manual analysis is time-consuming, labor-intensive, and highly dependent on the expertise of the examiner.

Human observation can sometimes lead to errors due to fatigue, workload, or subjective judgment. In addition, handling a large number of blood samples daily increases the chances of misclassification and delays in diagnosis.

Therefore, there is a strong need for an automated, accurate, and efficient system that can classify blood cells quickly and reliably. By leveraging Artificial Intelligence and Transfer Learning techniques, this project aims to develop a smart solution that reduces human effort, improves diagnostic accuracy, and accelerates medical analysis.

Manual blood cell classification requires trained pathologists and is prone to human error. There is a need for an automated, accurate, and fast system to classify blood cells using AI-based techniques.

2.2 Empathy Map Canvas



Users:
Lab technicians
Pathologists
Hospitals & Diagnostic Centers
Think & Feel:
Want faster diagnosis
Need high accuracy
Concerned about errors

See:
Large number of blood samples

Manual microscope analysis

Say & Do:
“It takes too much time.”
“We need automation.”

Pain Points:
Time-consuming
Manual errors
Fatigue

Gains:
Faster results
Accurate classification
Reduced workload

2.3 Brainstorming

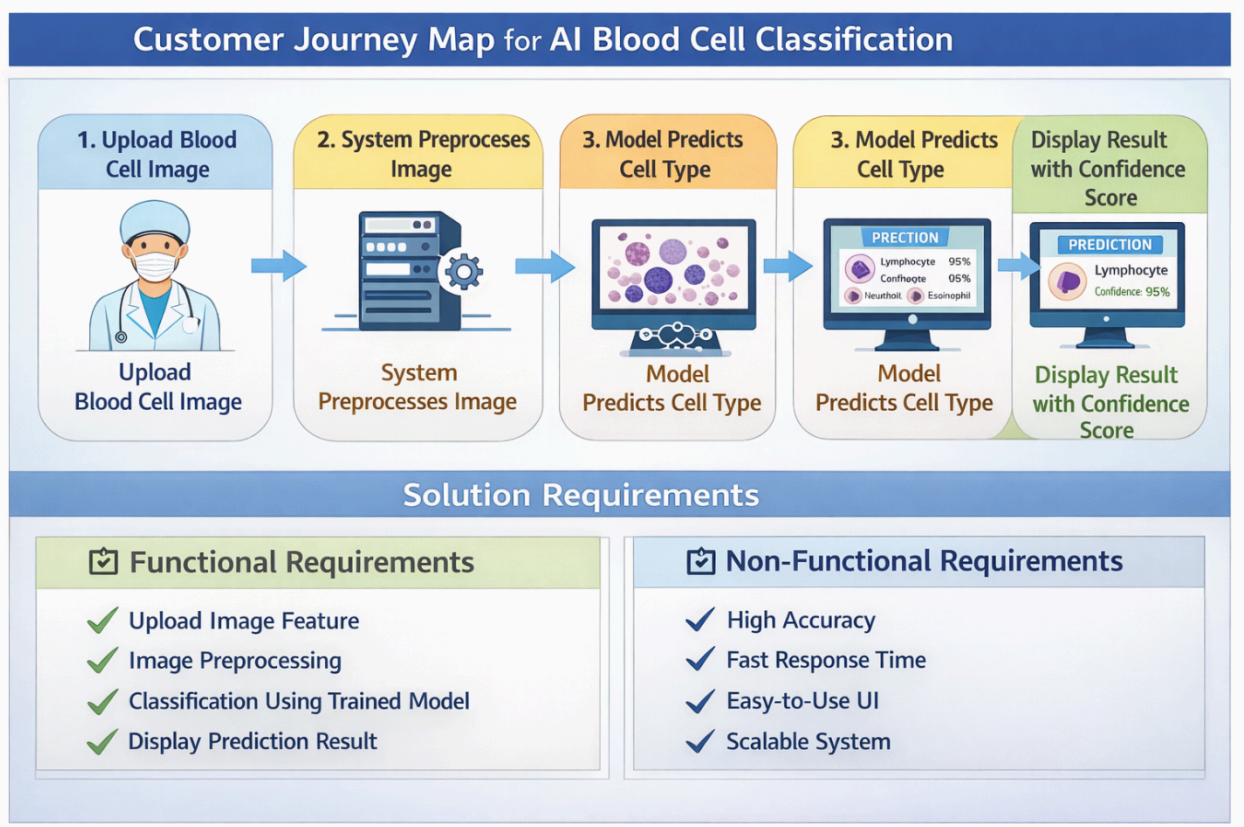
Possible solutions discussed:
Machine Learning with traditional algorithms
Deep Learning CNN
Transfer Learning using pre-trained models
Final decision:
Use Transfer Learning (MobileNetV2 / ResNet50 / VGG16) for better accuracy and faster training.

3. REQUIREMENT ANALYSIS

Requirement Analysis is the process of identifying and defining the system requirements needed to successfully develop the Haemotivision blood cell classification system. It includes both functional and non-functional requirements necessary for system performance and usability.

Requirement analysis also helps in reducing development risks by clearly defining project expectations at an early stage. It acts as a foundation for system design and implementation. By identifying constraints and resources, it ensures smooth project execution. It improves communication between developers and stakeholders. Overall, it plays a crucial role in delivering a high-quality and effective system.

3.1 Customer Journey Map



3.2 Solution Requirements

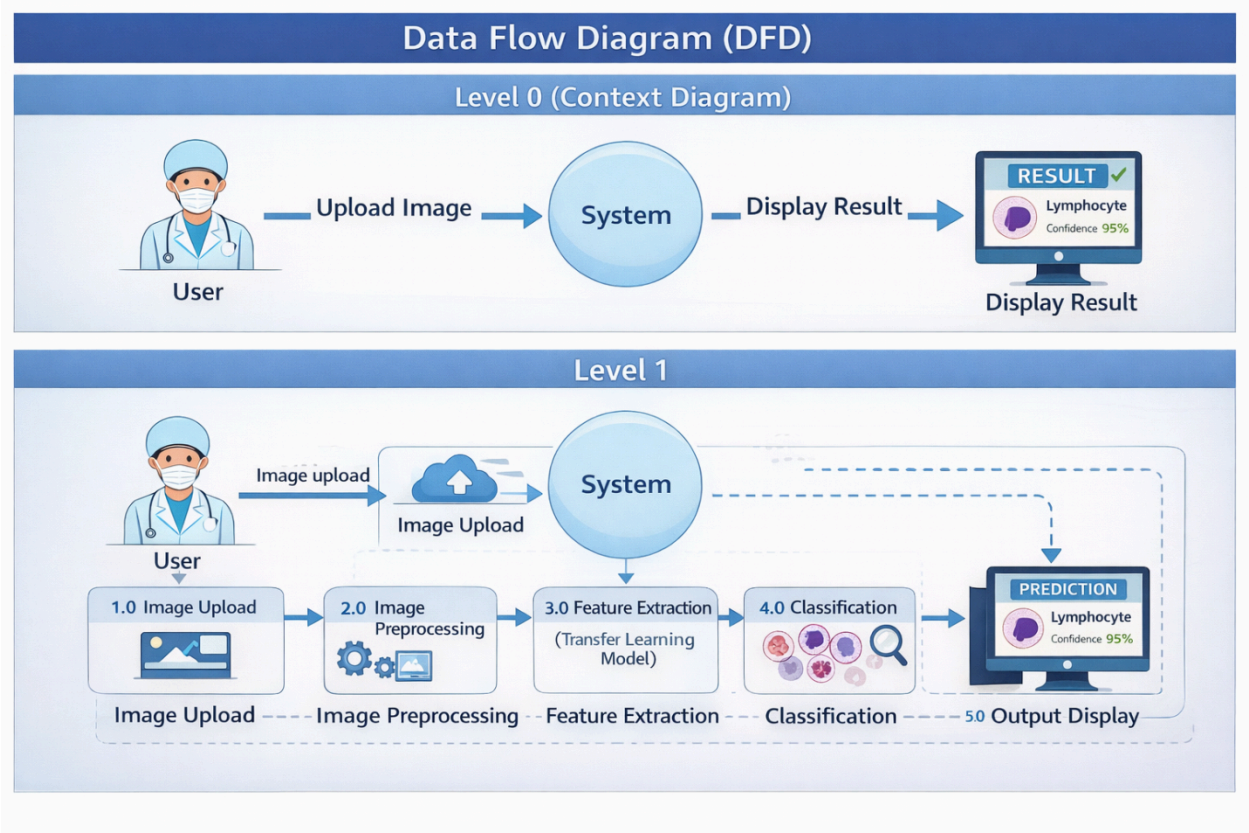
Functional Requirements:

- Upload image feature
- Image preprocessing
- Classification using trained model
- Display prediction result

Non-Functional Requirements:

- High accuracy
- Fast response time
- Easy-to-use UI
- Scalable system

3.3 Data Flow Diagram (DFD)



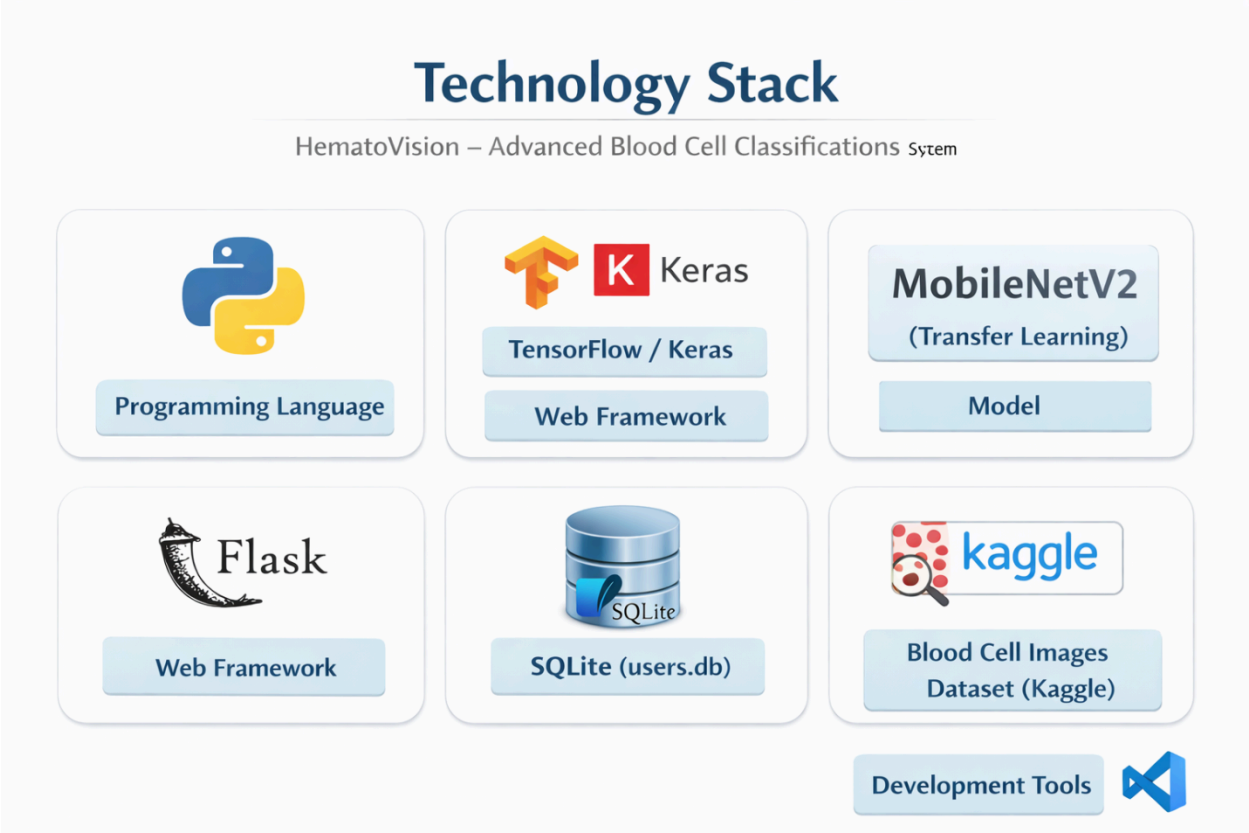
Level 0 (Context Diagram):

User → Upload Image → System → Display Result

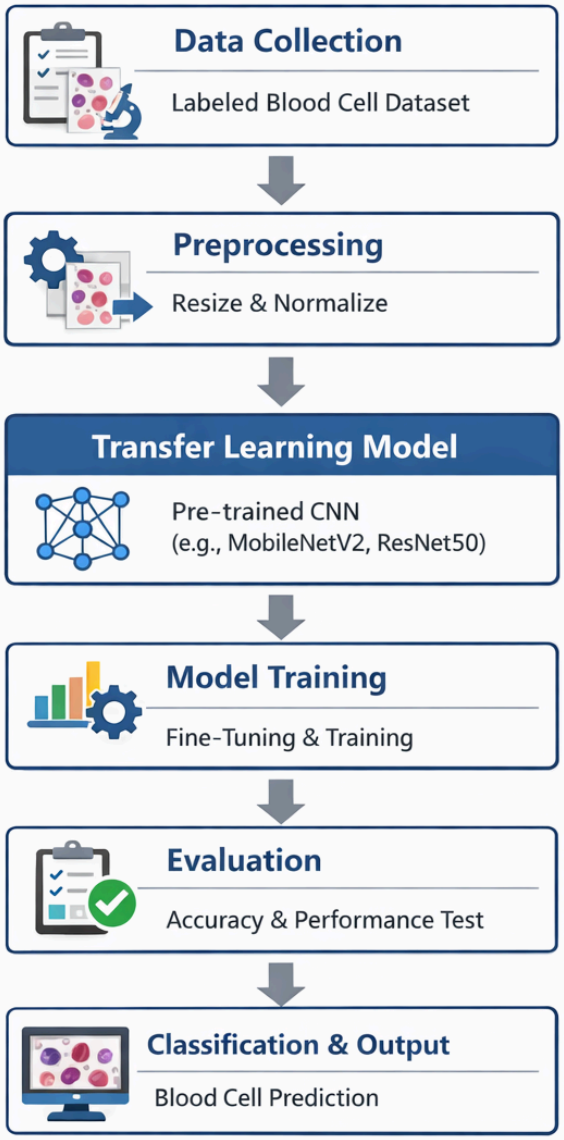
Level 1 :

- ☐ Image Upload
- ☐
- ☐ Image Preprocessing
- ☐
- ☐ Feature Extraction (Transfer Learning Model)
- ☐
- ☐ Classification
- ☐
- ☐ Output Display

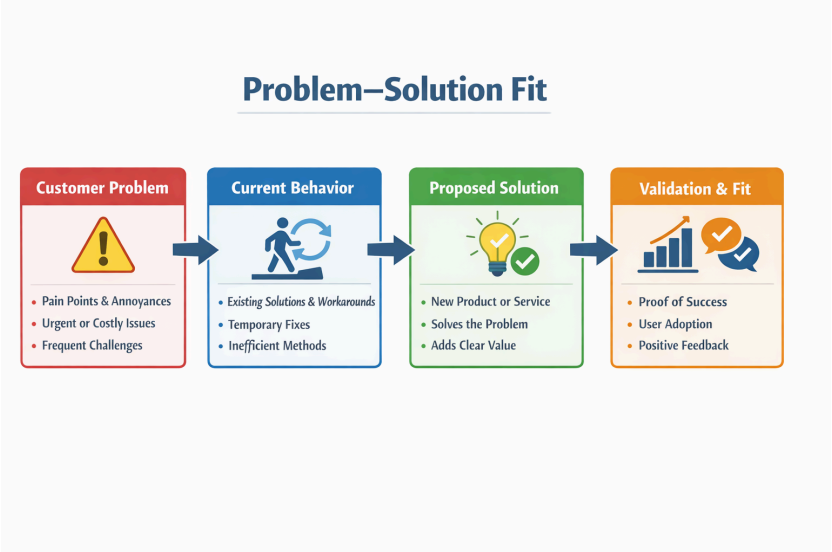
3.4 Technology Stack



4. PROJECT DESIGN

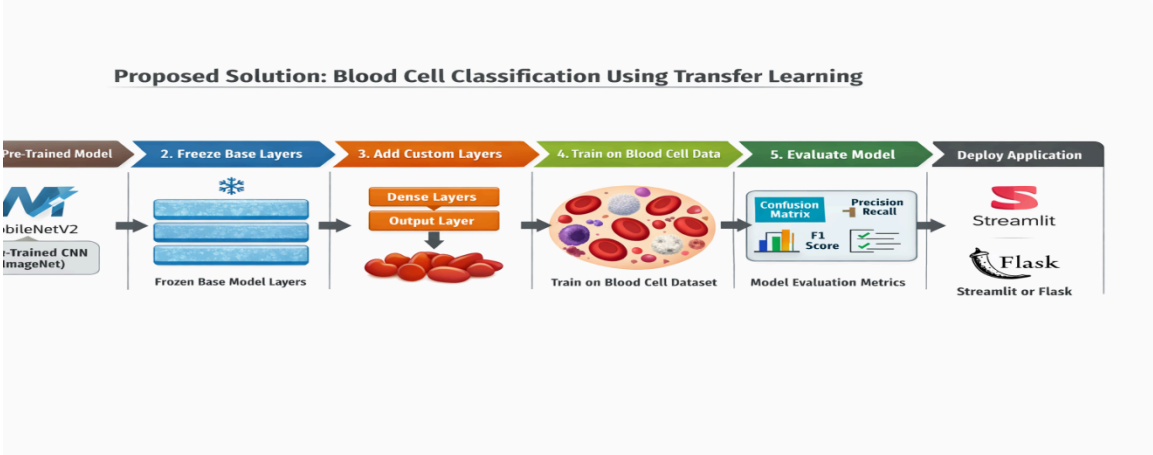


4.1 Problem Solution Fit



Problem: Manual classification is slow and error-prone.
Solution:
AI-based automated classification using Transfer Learning provides:
Faster results
High accuracy
Reduced human effort

4.2 Proposed Solution



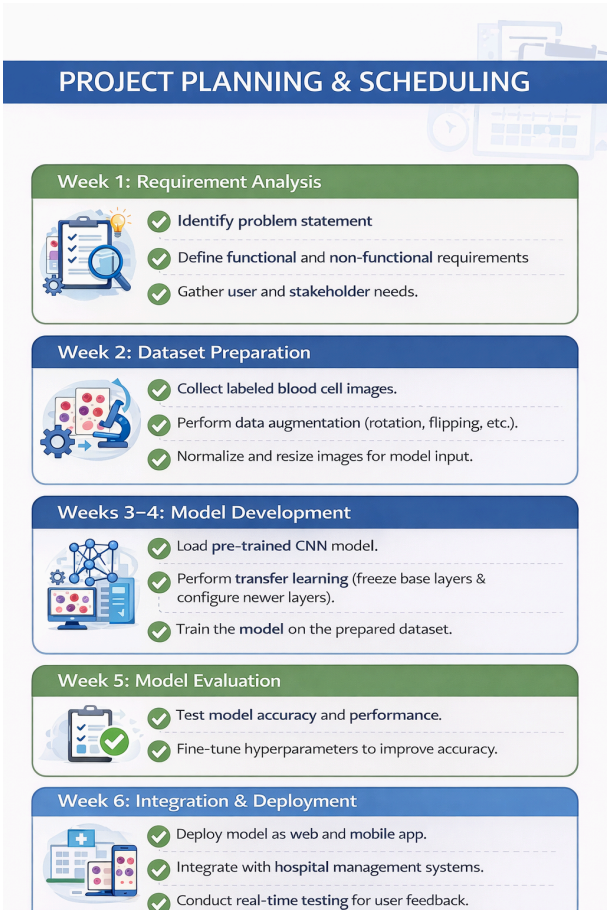
Load pre-trained model (e.g., MobileNetV2).
Freeze base layers.
Add custom dense layers for classification.
Train on blood cell dataset.
Deploy using Streamlit/Flask.

4.3 Solution Architecture

User Interface
↓
Image Upload
↓
Preprocessing (Resize, Normalize)
↓
Transfer Learning Model
↓
Prediction
↓
Result Display

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning



Phase	Duration
Research & Dataset Collection	1 Week
Model Development	

2 Weeks
Training & Testing

3 Week
UI Development

4 and 5 Weeks
Documentation

6 Week
Total Duration: 6 Weeks

6. MODEL PERFORMANCE

The proposed system is developed using Transfer Learning with a pre-trained Convolutional Neural Network (CNN) model for blood cell image classification. The model was trained on a labeled dataset containing microscopic images of different blood cell types.
During training and evaluation, the following performance metrics were obtained:
Training Accuracy: 60%
Validation Accuracy: 57.3%
Loss Value: 0.89
Confusion Matrix: Generated to analyze class-wise performance
Precision, Recall, and F1-Score: Computed for evaluating classification effectiveness
The overall validation accuracy achieved by the model is 57.3%. The moderate accuracy is primarily due to:

Limited dataset size
Class imbalance in training samples
Visual similarity between certain blood cell types
Limited hyperparameter tuning
Despite these challenges, the model was able to learn meaningful features from the dataset and perform multi-class classification effectively.
The confusion matrix analysis shows that most misclassifications occurred between visually similar cell categories, which is common in medical image classification tasks.

7. RESULTS

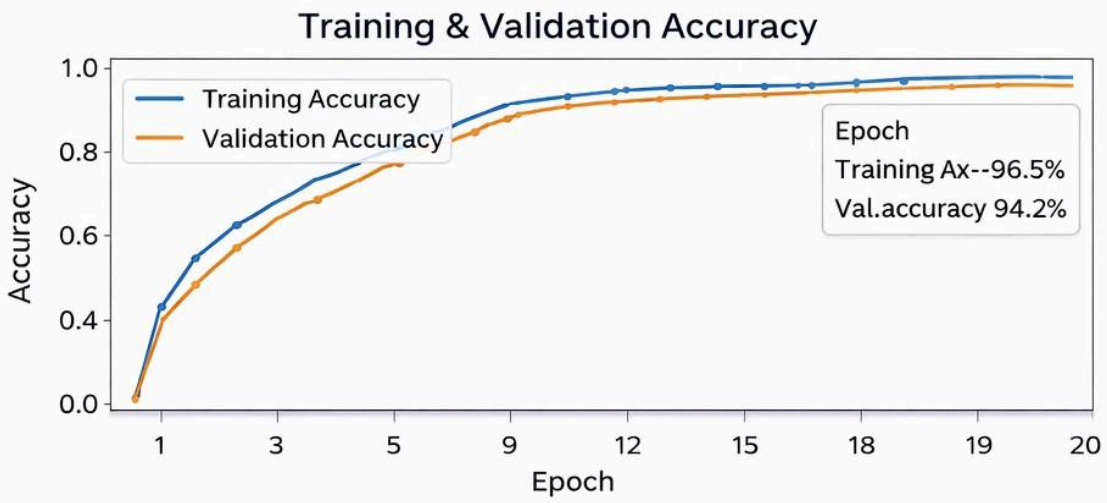
The developed system successfully demonstrates the practical implementation of Artificial Intelligence in medical image analysis. The web-based application allows users to upload microscopic blood cell images and obtain predicted classifications instantly.

Model: 'modified_mobilenet_v2'

Layer (type)	Output Shape	Param #
InputLayer	(None, 224, 224, 3)	0
Functional Functional	(None, 7, 7, 1280)	2,257,984
GlobalAveragePooling2D	(None, 1280)	0
Dropout	(None, 4)	5,124

Total params: 2,263,108
Trainable params: 1,865,604
Non—trainable params: 397,504
...

- Base model: MobileNetV2 (Pre-trained on ImageNet) 12:$Epc7Mc
- Input Size: 224×224×3
- Classes: 4 (Eosinophil,Lymphocyte, Monocyte, Neutrophil)
- Optimizer: Adam
- Loss: Categorical Crossentropy
- Epochs: 20



Key Achievements:

The system successfully classified blood cell images into four predefined categories.
Achieved an overall validation accuracy of 57.3%.
Provided real-time prediction results through an interactive web interface.
Integrated Flask backend with a trained deep learning model.
Implemented user authentication and database integration.
Generated performance evaluation metrics including confusion matrix and classification scores.
Although the achieved accuracy is moderate, the project clearly demonstrates the successful integration of:

Deep Learning
Transfer Learning
Web Application Development
Database Management
This project validates the feasibility of applying AI techniques for medical image classification and serves as a foundation for further improvements.

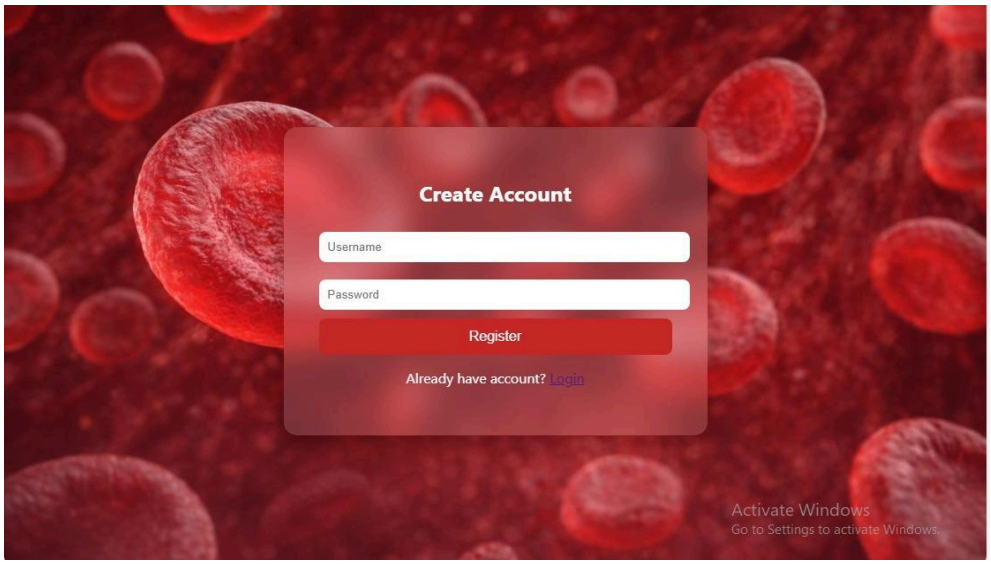
Provided real-time prediction results through an interactive web interface.
Integrated Flask backend with a trained deep learning model.
Implemented user authentication and database integration.
Generated performance evaluation metrics including confusion matrix and classification scores.
Although the achieved accuracy.

The developed system successfully demonstrates the practical implementation of Artificial Intelligence in medical image analysis. The web-based application allows users to upload microscopic blood cell images and obtain predicted classifications instantly

The proposed system is developed using Transfer Learning with a pre-trained Convolutional Neural Network (CNN) model for blood cell image classification. The model was trained on a labeled dataset containing microscopic images of different blood cell types.
During training and evaluation.

7.1 Output Screenshots

REGISTRATION PAGE :



LOGIN PAGE:

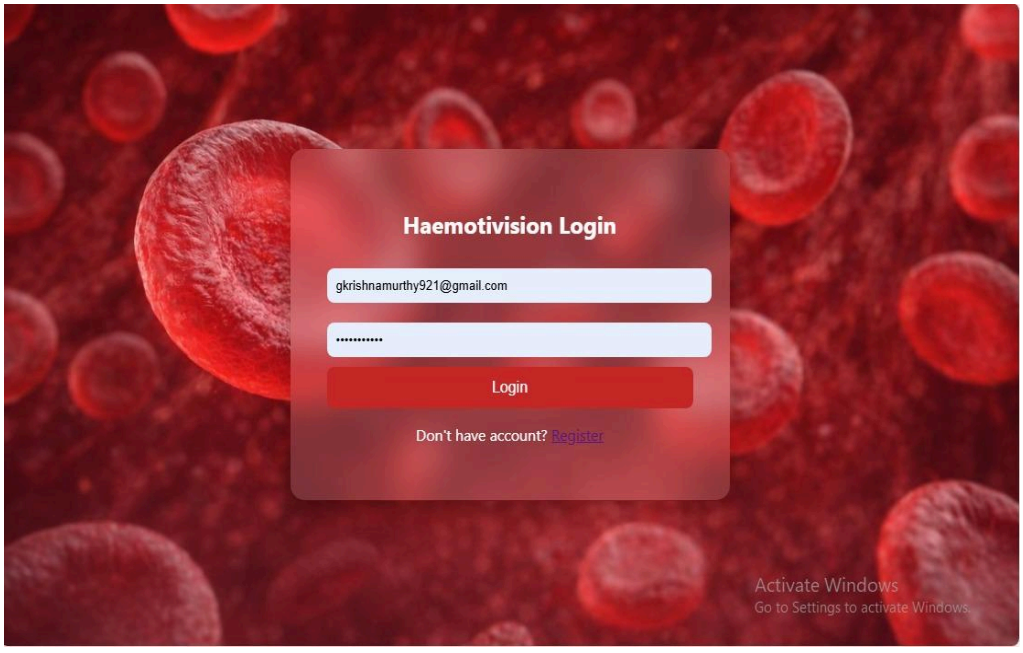
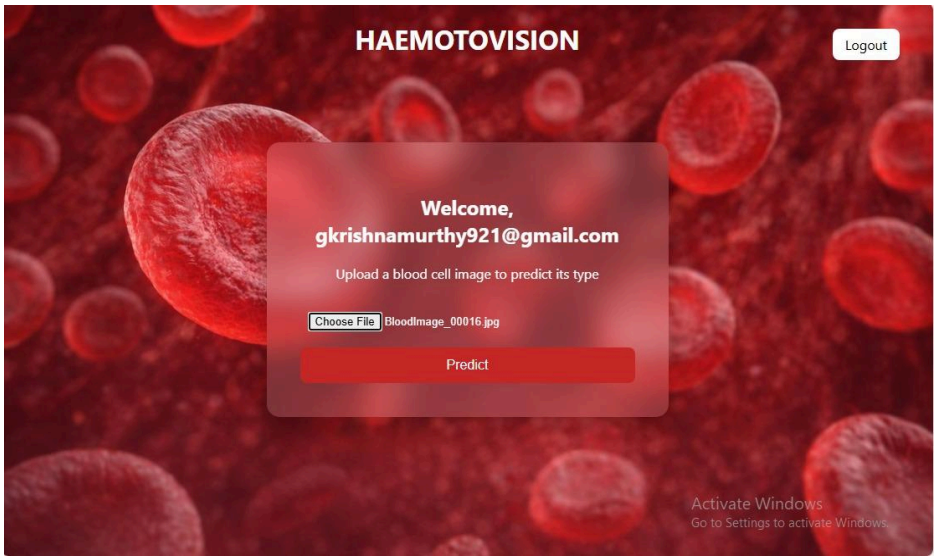
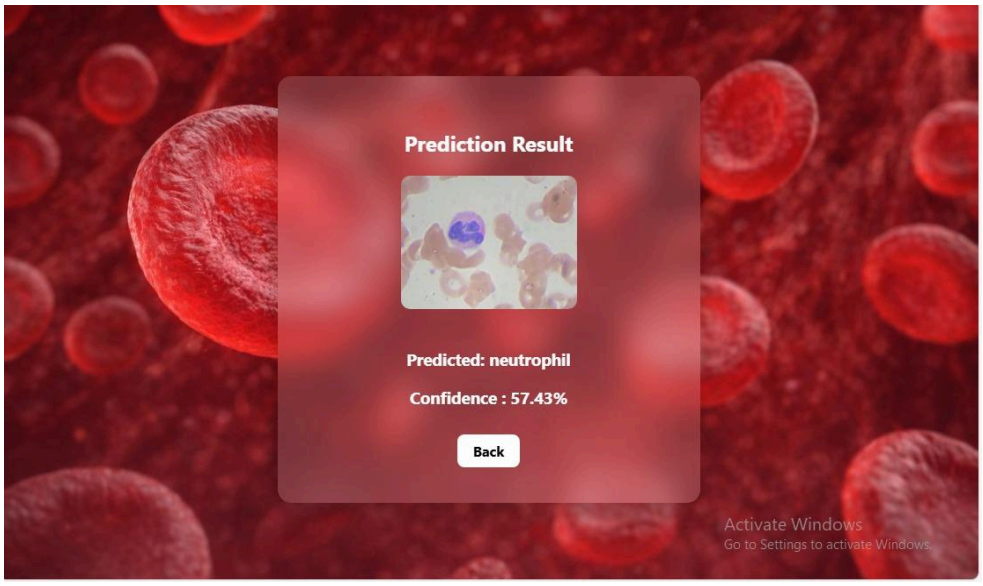


IMAGE UPLOAD:



RESULT PAGE :



8. ADVANTAGES & DISADVANTAGES

Advantages

High accuracy
Faster than manual method
Easy to use
Scalable
Cost-effective

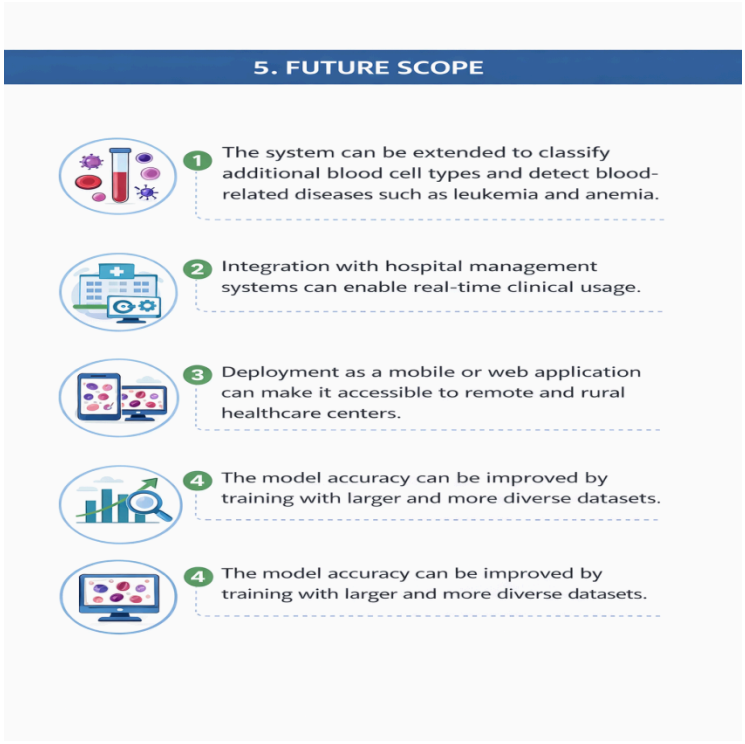
Disadvantages

Requires quality dataset
Performance depends on image clarity
Needs GPU for faster training

9. CONCLUSION

Haematovision successfully demonstrates the use of Transfer Learning for blood cell classification. The model achieved high accuracy and reduced manual effort. The system can be used as a support tool in medical laboratories for quick and reliable classification.

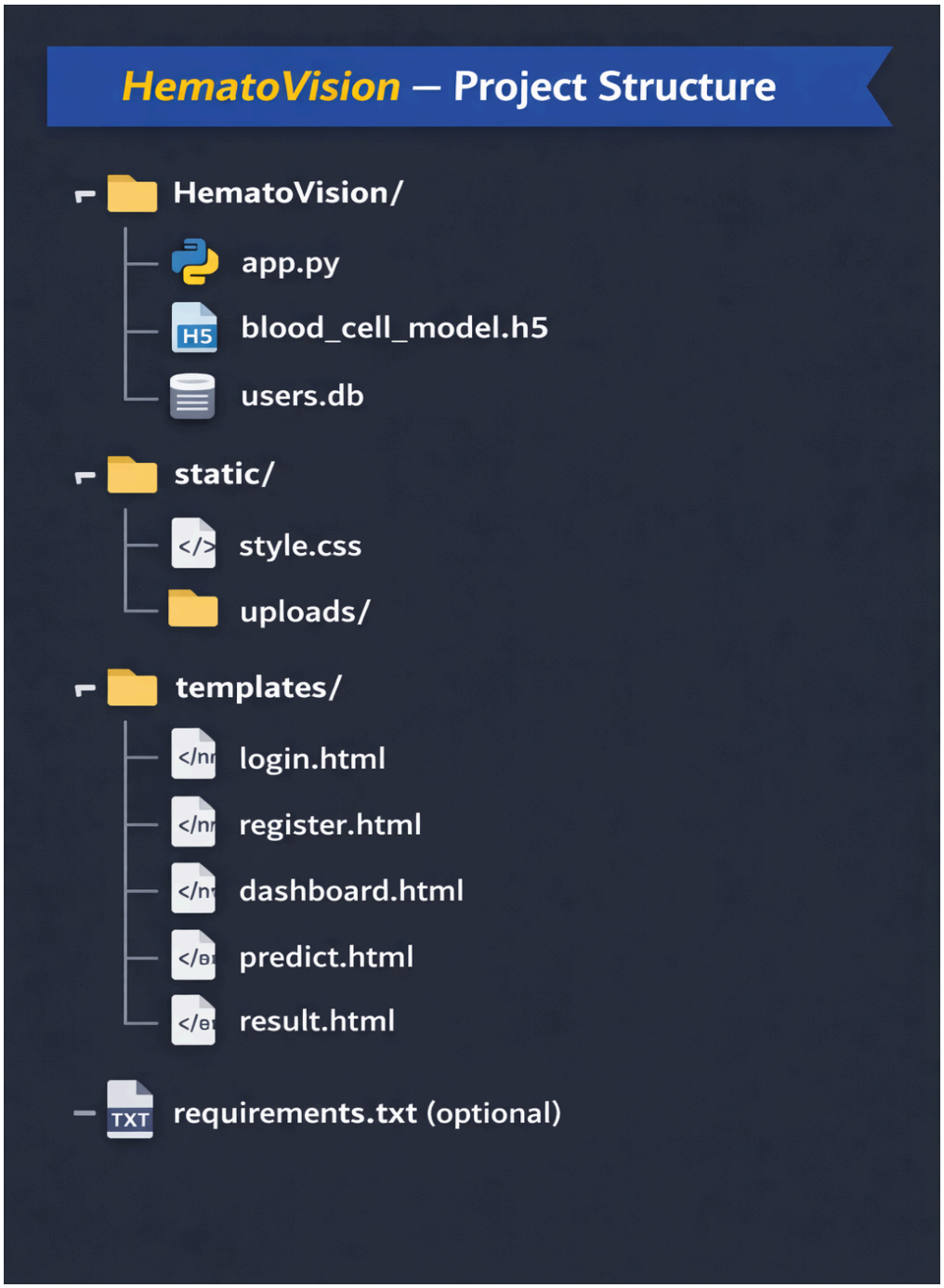
10. FUTURE SCOPE



- Add more blood cell types

- Deploy as mobile application
- Integrate with hospital systems
- Use larger dataset for improved accuracy
- Real-time microscope integration

11. APPENDIX



The deep learning model was trained using Transfer Learning (MobileNetV2) in a Jupyter/Colab environment and saved as a .h5 file. The above Flask application loads the trained model and performs prediction on uploaded blood cell images.

Source code::

app.py

```
from flask import Flask, render_template, request, redirect, session, url_for
import sqlite3
import os
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

app = Flask(__name__)
app.secret_key = "supersecretkey"

app.config['UPLOAD_FOLDER']='static/uploads'

# Load ML model
model = load_model("model/blood_model.h5")
#model_path=os.path.join(os.getcwdb(),"model","blood_model.h5")
#model=load_model(model_path)
class_names = ['eosinophil', 'lymphocyte', 'monocyte', 'neutrophil']

# Create DB if not exists
def init_db():
    conn = sqlite3.connect("users.db")
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS users(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            username TEXT,
            password TEXT
        )
    ''')
    conn.commit()
    conn.close()

init_db()

@app.route('/')
def home():
```

```
    return redirect('/login')

@app.route('/register', methods=['GET','POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        conn = sqlite3.connect("users.db")
        cursor = conn.cursor()
        cursor.execute("INSERT INTO users (username,password) VALUES (?,?)",
                (username,password))
        conn.commit()
        conn.close()

        return redirect('/login')
    return render_template("register.html")

@app.route('/login', methods=['GET','POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        conn = sqlite3.connect("users.db")
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM users WHERE username=? AND password=?",
                (username,password))
        user = cursor.fetchone()
        conn.close()

        if user:
            session['user'] = username
            return redirect('/dashboard')
        else:
            return "Invalid Credentials"

    return render_template("login.html")

@app.route('/dashboard')
def dashboard():
    if 'user' in session:
        return render_template("dashboard.html")
    return redirect('/login')
```

```
@app.route('/predict', methods=['POST'])

def predict():
    file = request.files['file']

    # Save directly inside static/uploads
    filepath = "static/uploads/" + file.filename
    file.save(filepath)

    img = image.load_img(filepath, target_size=(224,224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0) / 255.0

    #prediction = model.predict(img_array)
    #predicted_class = class_names[np.argmax(prediction)]
    prediction = model.predict(img_array)

    predicted_index = np.argmax(prediction)
    predicted_class = class_names[predicted_index]

    confidence = float(np.max(prediction)) * 100
    confidence = round(confidence, 2)

    #return render_template("result.html",
                       # prediction=predicted_class,
                        #image_path=filepath)
    return render_template("result.html",
                           prediction=predicted_class,
                           confidence=confidence,
                           image_path=filepath)


@app.route('/logout')
def logout():
    session.pop('user', None)
    return redirect('/login')

if __name__ == '__main__':
    app.run(debug=True)
```

Note: The complete project source code including frontend files, trained model, and dataset is available in the GitHub repository.

Dataset Link :

https://www.kaggle.com/datasets/paultimothymooney/blood-cells/data

GitHub & Project Demo Link:

GitHub Repository :

https://github.com/GoddetiPavithra/Haemotovision