Artificail Intelligence&Machine learning

Project Report

HematoVision

Advanced Blood Cell Classification Using Transfer Learning

1.Introduction:

1.1Problem statement:

HematoVision aims to develop an accurate and efficient model for classifying blood cells by employing transfer learning techniques. Utilizing a dataset of 12,000 annotated blood cell images, categorized into distinct classes such as eosinophils, lymphocytes, monocytes, and neutrophils, the project leverages pre-trained convolutional neural networks (CNNs) to expedite training and improve classification accuracy. Transfer learning allows the model to benefit from pre-existing knowledge of image features, significantly enhancing its performance and reducing computational costs. This approach provides a reliable and scalable tool for pathologists and healthcare professionals, ensuring precise and efficient blood cell classification.

Objective:

- Automate classification of different blood cell types (e.g., RBCs, WBCs, platelets).
- Use transfer learning to overcome data scarcity and reduce training time.
- Compare different pretrained models (ResNet, DenseNet, EfficientNet, ViT).
- Improve model interpretability using explainable AI (Grad-CAM, LIME).

1.2 Proposed solution:

HematoVision aims to address this gap by leveraging transfer learning and advanced neural networks to develop a robust, interpretable, and scalable blood cell classification system suitable for clinical and remote diagnostic settings.

Targeted users:

- Pathologists & Hematologists
- Medical Laboratory Technicians
- Clinicians in Rural/Resource-Limited Settings
- Medical Students & Educators
- Hospitals & Diagnostic Centers
- Biomedical AI Researchers & Developers
- HealthTech Companies & NGOs

Excepted outcome:

- High-Accuracy Blood Cell Classification
- Efficient Use of Limited Data
- Improved Diagnostic Speed and Consistency
- Explainable AI Integration
- Educational Tool for Medical Training
- Foundation for Future Research & Innovation

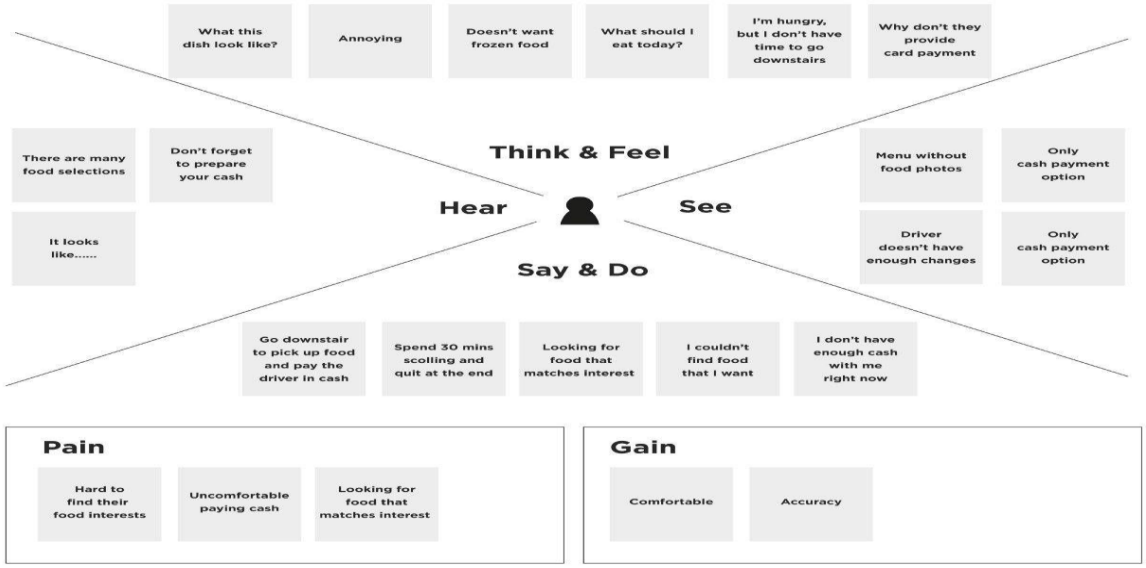
2.Ideation phase:

2.1 Problem statement:

HematoVision aims to develop an accurate and efficient model for classifying blood cells by employing transfer learning techniques. Utilizing a dataset of 12,000 annotated blood cell images, categorized into distinct classes such as eosinophils, lymphocytes, monocytes, and neutrophils, the project leverages pre-trained convolutional neural networks (CNNs) to expedite training and improve classification accuracy. Transfer learning allows the model to benefit from pre-existing knowledge of image features, significantly enhancing its performance and reducing computational costs. This approach provides a reliable and scalable tool for pathologists and healthcare professionals, ensuring precise and efficient blood cell classification.

2.2 Empathy map canvas:

Empathy Map Canvas for HematoVision: Advanced Blood Cell Classification, tailored primarily for the target user — pathologists, lab technicians, medical students, and healthcare providers using or interacting with the system.



2.3 Brainstorming:

1. Problem Understanding

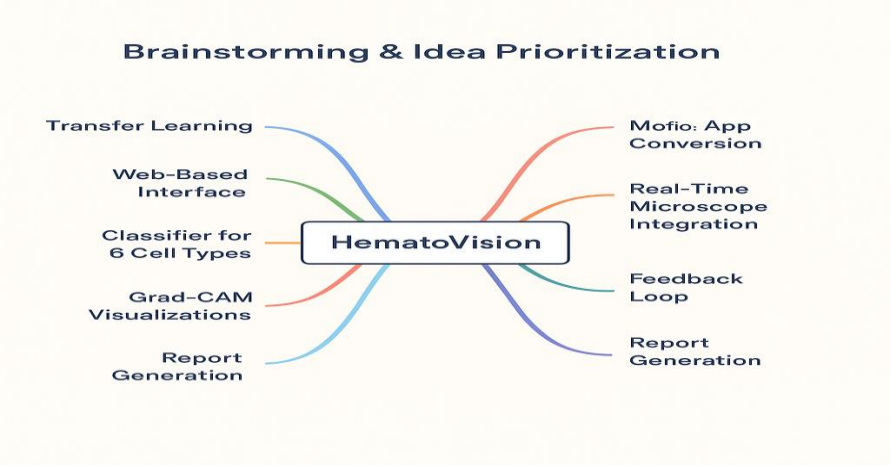
- Manual classification is time-consuming and error-prone.
- Need for quick, reliable diagnosis in hospitals and rural labs.
- Can deep learning assist medical professionals?

2. Solution Ideas

Idea	Description
Use Transfer Learning	Use pre-trained models like EfficientNet or ResNet.
Build Web App	User-friendly Flask UI for uploading images.
Focus on 6 Blood Cell Types	RBC, Platelet, Neutrophil, Lymphocyte, Monocyte, Eosinophil.
Auto-Predict with Confidence	Return class + confidence %.
Exportable Reports	Enable saving results as PDF/CSV.
Mobile Version	Use TensorFlow Lite to build an app for rural health workers.
Grad-CAM Heatmaps	Visualize prediction focus areas for transparency.
API Endpoint	Allow hospitals to send images via REST API.
Feedback System	Let users correct wrong predictions to improve the model.
Real-Time Classification	Direct input from digital microscope cameras.

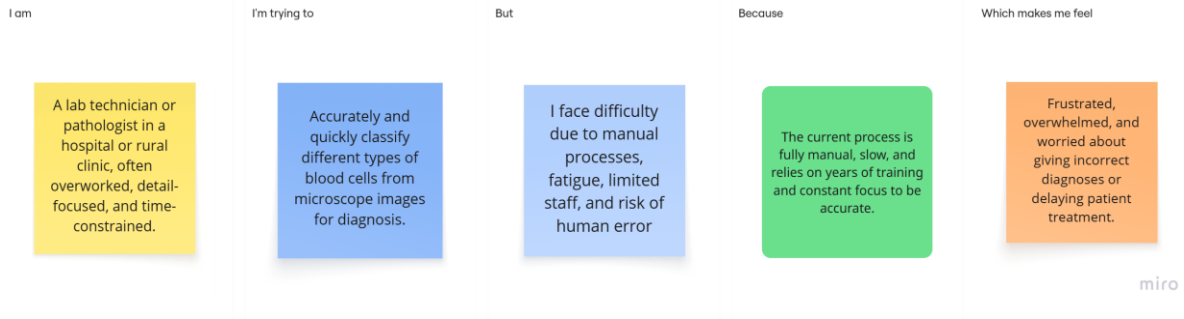
3. User-Centered Features

- Support for high-resolution microscope images
- Simple upload and click-based UI
- Support for edge deployment (low power devices)
- Multi-language or local-language interface (for rural use)



3. REQUIREMENT ANALYSIS:

3.1 Customer Journey map



Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	a lab technician or pathologist	classify blood cells quickly and accurately	it's time-consuming and error-prone	the process is manual and requires constant focus	Frustrated, overwhelmed, and worried about giving incorrect diagnoses or delaying patient treatment.

3.2 Solution Requirement

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form

		Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Interface (UI)	Allows users to upload blood smear images (JPG/PNG). Displays predicted blood cell type and confidence level.
FR-4	Image Preprocessing Module	Automatically resizes and normalizes input images. Converts images to suitable input format for the model
FR-5	Prediction Output Module	Displays prediction results with confidence score. Optionally shows visual markers/highlights on cell image (futur
FR-6	Data Validation	Ensures only valid image files are uploaded. Handles error messages for unsupported formats or failed uploads.
FR-7	Model Management	Supports updating or replacing the deep learning model file without changing the core code
FR-8	Security & Privacy	Ensures uploaded images are not stored permanently unless explicitly allowed. Follows basic privacy compliance for patient data (if used in real-time clinical settings).

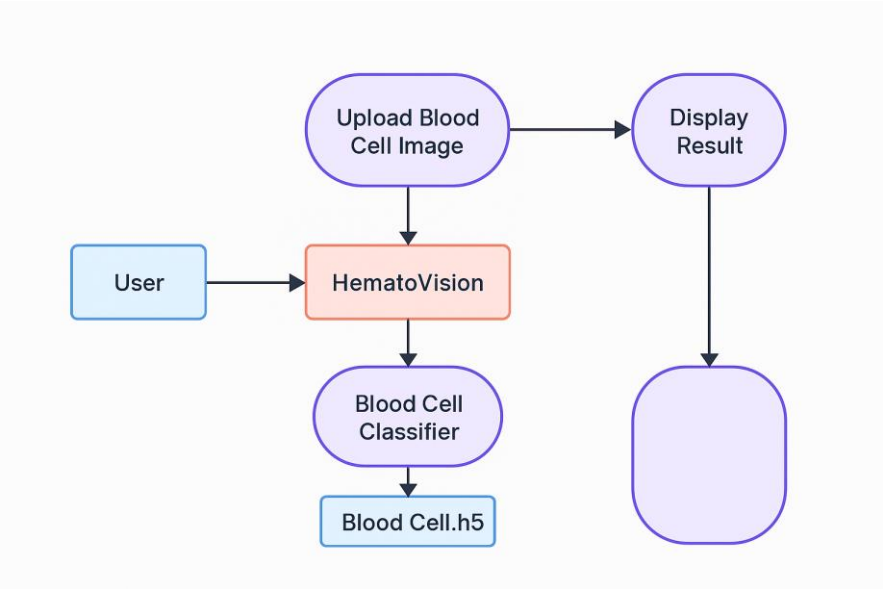
Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

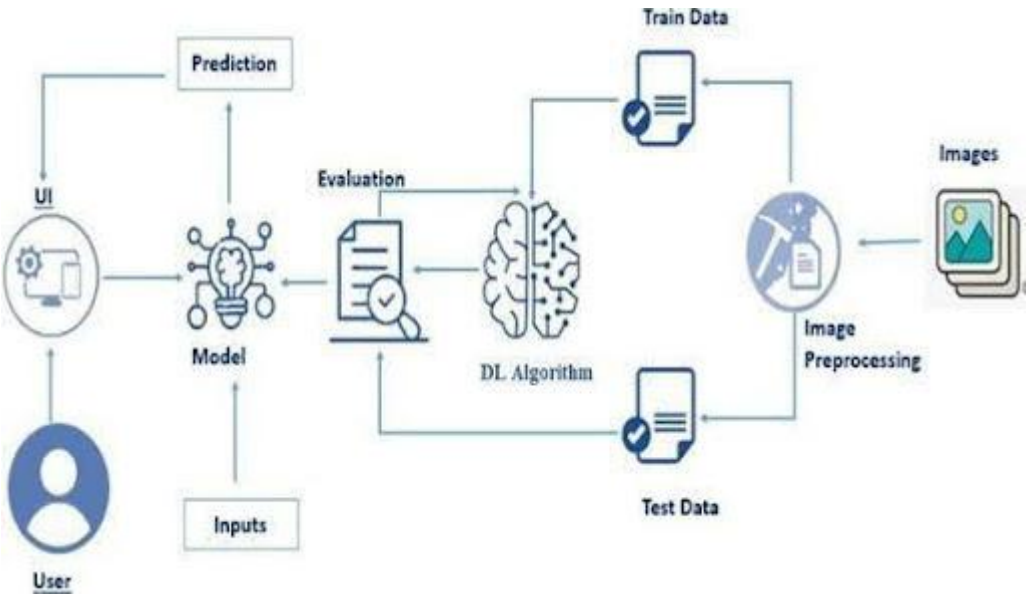
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Interface should be simple and intuitive for non-technical users such as lab technicians. Minimal training should be needed to operate the system.
NFR-2	**Security**	Image uploads should be processed securely and deleted after prediction unless storage is required.No personally identifiable information (PII) should be stored without consent.
NFR-3	**Reliability**	The system should ensure high uptime when deployed on cloud platforms. Predictions should remain consistent across repeated evaluations of the same image
NFR-4	**Performance**	The system should deliver blood cell classification results within 5 seconds for each image upload. The model should maintain at least 90% accuracy on test data.
NFR-5	**Portability**	The solution should run on **multiple platforms**—locally (Windows/Linux) and online (via web deployment). Future deployment on **mobile devices** via TensorFlow Lite should be supported.
NFR-6	**Scalability**	The application should support scaling to handle multiple users simultaneously (when deployed online).Future upgrades should allow classification of additional cell types or diseases.

3.3 Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



3.4 Technology stack:



S.No	Component	Description	Technology
1	Data Collection	Collect 12,000 labeled blood cell images (eosinophils, lymphocytes, etc.)	Image Datasets, Kaggle, BCCD

2	Data Preprocessing	Resize, normalize, augment images before training	NumPy, OpenCV, TensorFlow ImageDataGen
3	Model Architecture	Load pre-trained CNN for feature extraction and classification	Transfer Learning, ResNet50 (Keras)
4	Model Training	Train the classifier on blood cell dataset using fine-tuning	TensorFlow, Keras
5	Prediction Module	Predict cell type with confidence score from uploaded image	NumPy, TensorFlow, Softmax
6	Flask Backend	Handles HTTP requests, file uploads, and prediction logic	Flask (Python Web Framework)
7	Frontend UI	Allows users to upload image and view prediction result	HTML, CSS, Jinja2 (Flask Templating)
8	Static File Handling	Saves uploaded images to server and serves them back to user	Flask static/ directory
9	Output Display	Shows predicted label, confidence %, and image preview	HTML, Flask response rendering
10	Platform Support	Runs locally (CMD, PyCharm) or deploys online (Streamlit/Render)	Localhost, Render, Streamlit Cloud

4. PROJECT DESIGN:

4.1 Problem Solution Fit:

Manual microscopic analysis of blood cells is a time-consuming and error-prone process that relies heavily on the expertise and availability of trained pathologists. In many rural and resource-limited healthcare settings, there is a shortage of skilled technicians, which further delays accurate diagnosis and patient care. Additionally, even experienced professionals can misclassify cells due to fatigue, leading to inconsistent results. The traditional approach is not scalable, particularly in emergency scenarios or during large-scale testing when speed and reliability are critical.

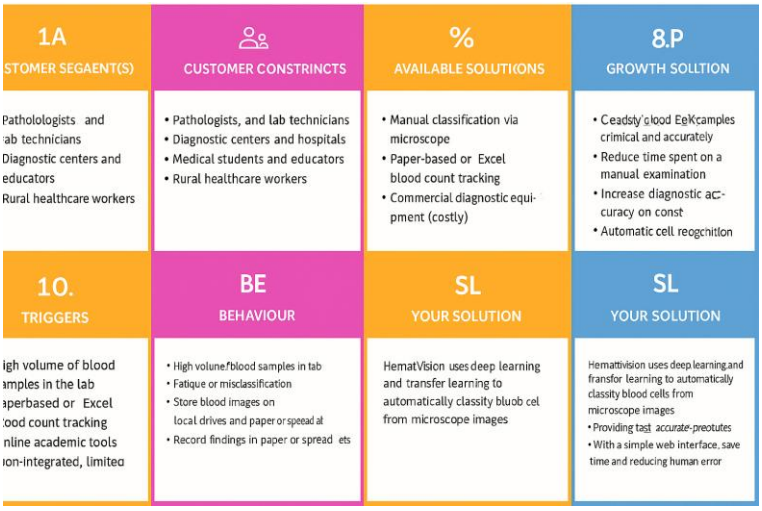
HematoVision addresses these challenges by offering an AI-powered blood cell classification system built using transfer learning techniques. Leveraging pre-trained models like EfficientNet, the system quickly and accurately identifies various blood cell types, including red blood cells, white blood cells, and platelets. With a simple and user-friendly web

interface built on Flask, the tool enables even non-experts to upload microscope images and receive predictions with high confidence in seconds. This significantly reduces the time and effort required for diagnosis, ensures consistency, and scales efficiently without the need for expensive equipment or extensive training. HematoVision presents a practical, cost-effective, and scalable solution that fits seamlessly into modern laboratory workflows and can transform hematology diagnostics, especially in underserved regions.

Purpose:

The purpose of the HematoVision project is to develop an AI-powered system that can automatically classify blood cells from microscope images using transfer learning techniques. This project aims to assist medical professionals—especially in resource-constrained environments—by providing a fast, accurate, and consistent method to identify blood cell types such as red blood cells, white blood cells, and platelets.

By integrating deep learning with a user-friendly web interface, HematoVision seeks to reduce dependency on manual analysis, minimize human error, and accelerate diagnostic workflows in clinical laboratories. The project also promotes accessible and cost-effective healthcare technology by making advanced image classification usable even in rural and remote settings.



4.2 Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	HematoVision aims to solve these challenges by developing an AI-powered system that uses transfer learning with pre-trained CNNs to classify blood cells accurately and efficiently. This system

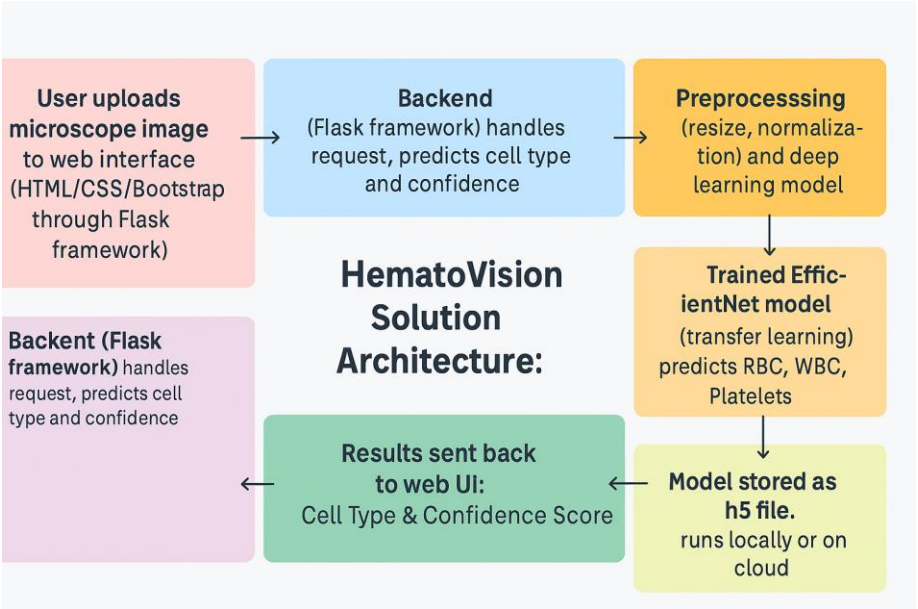
		minimizes diagnostic delays, reduces human error, and provides scalable diagnostic support, especially in telemedicine and educational contexts.
2.	Idea / Solution description	HematoVision is an AI-powered solution designed to automate the classification of blood cells from microscope images using advanced transfer learning techniques. The core idea is to leverage pre-trained convolutional neural network (CNN) models—such as EfficientNet or ResNet—and fine-tune them with a dataset of labeled blood cell images to achieve high classification accuracy. These models can identify and distinguish between different types of blood cells, including Red Blood Cells (RBCs), White Blood Cells (WBCs), and Platelets.
3.	Novelty / Uniqueness	HematoVision is unique in combining deep learning accuracy with real-world usability. Unlike traditional diagnostic tools, it leverages transfer learning with models like EfficientNet to achieve fast and precise blood cell classification even with limited data. Its integration into a simple web interface makes it accessible to non-experts, including lab staff in rural areas. The project stands out for its scalability, low-cost deployment, and future potential for explainability (Grad-CAM) and mobile health applications, making it a novel and practical AI solution for digital hematology.
4.	Social Impact / Customer Satisfaction	HematoVision empowers healthcare systems by making accurate blood cell classification faster, more accessible, and cost-effective—especially in rural and

		underserved areas. By reducing the burden on pathologists and minimizing diagnostic errors, it improves patient outcomes and speeds up treatment decisions. The user-friendly interface ensures that even non-experts can operate the system with ease, leading to high customer satisfaction. HematoVision contributes to equitable healthcare access, supporting national goals of digital and AI-integrated medical services.
5.	Business Model (Revenue Model)	**HematoVision** follows a **B2B (Business-to-Business)** model targeting clinics, diagnostic labs, and healthcare institutions. Revenue can be generated through multiple channels: • **Software-as-a-Service (SaaS):** Monthly or yearly subscription for access to the web-based blood cell classification tool. • **One-Time Licensing:** Hospitals or labs can purchase a lifetime license for offline or on-premise deployment. • **Pay-per-Use Model:** Small clinics or remote centers can pay per image processed, making it affordable and scalable. • **Mobile App (Future Scope):** A freemium mobile version with basic features, and premium options for AI reporting and offline use. • **Customization Services:** Revenue from personalized integrations with hospital management systems (HMS) or digital microscopes.

6.	Scalability of the Solution	**HematoVision** is designed to be highly scalable across both technical and geographical dimensions. Technically, the use of **transfer learning** allows the model to be trained or fine-tuned with new data, enabling expansion to more blood cell types or diseases (e.g., malaria, leukemia). The Flask-based web app can be **hosted on cloud platforms** for global access or **converted to a mobile app** using TensorFlow Lite for offline use in rural areas. the solution is **flexible, low-cost, and adaptable**, making it suitable for widespread deployment across diverse healthcare environments.

4.3 Solution Architecture:

The HematoVision system is designed using a modular architecture that combines AI model deployment, web-based interfacing, and image preprocessing for efficient blood cell classification.



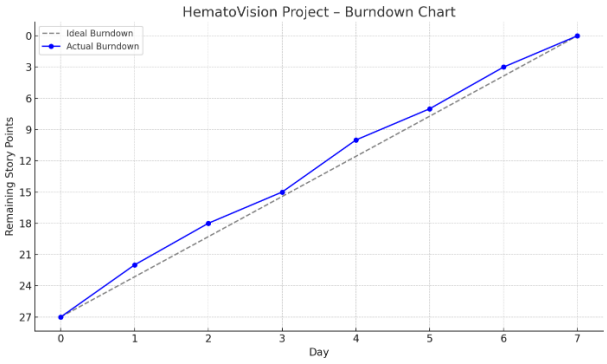
5. PROJECT PLANNING & SCHEDULING:

5.1 Project Planning:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High
Sprint-2		USN-3	As a user, I can register for the application through Facebook	2	Low
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	1	High
Sprint -3	Dashboard	USN-6	As a user, Register in github	2	Medium
Sprint -4		USN-7	As a user, Distributed project as tasks	1	Low
Sprint -2		USN-8	As a user, completed what I have the task	2	Medium
Sprint -3	Word	USN-9	As a user, completed the documentation as per the templates.	3	High
Sprint -3		USN-10	As a user, completed the demo video.	3	High
Sprint -3		USN-11	As a user, uploaded all files in github and shared the link .	2	medium

Burndown Chart:

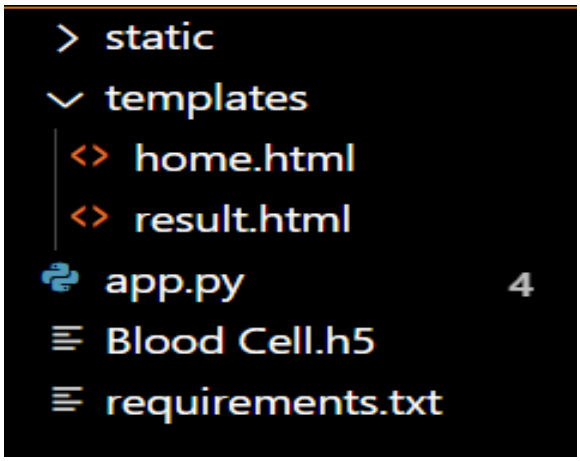
A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



6. FUNCTIONAL AND PERFORMANCE TESTING:

6.1 Performance Testing:

Project structure:



Templates :

Home.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Blood Cell Classifier</title>
</head>
<body>
    <h1>Upload a Blood Cell Image</h1>
    <form action="/predict" method="POST" enctype="multipart/form-data">
        <input type="file" name="file" required>
        <input type="submit" value="Predict">
    </form>
</body>
</html>
```

Result.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Prediction Result</title>
</head>
<body>
    <h1>Prediction Result</h1>
    <p><strong>Predicted Class:</strong> {{ prediction }}</p>
    <img src="{{ image_path }}" alt="Uploaded Image" width="300">
    <br><br>
    <a href="/">Try Another Image</a>
</body>
</html>
```

App.py :

```
from flask import Flask, render_template, request # type: ignore
from tensorflow.keras.models import load_model # type: ignore
from tensorflow.keras.preprocessing import image # type: ignore
import numpy as np # type: ignore
import os
app = Flask(__name__)
model = load_model("Blood Cell.h5")
# Define class labels (update based on your model's output classes)
class_names = ['EOSINOPHIL', 'LYMPHOCYTE', 'MONOCYTE', 'NEUTROPHIL']
@app.route('/')
def home():
    return render_template('home.html')
@app.route('/predict', methods=['POST'])
```

```
def predict():
    if 'file' not in request.files:
        return "No file uploaded", 400
    file = request.files['file']
    if file.filename == '':
        return "No selected file", 400
    if file:
        img_path = os.path.join('static', file.filename)
        file.save(img_path)
# Image preprocessing
        img = image.load_img(img_path, target_size=(64, 64))  # Adjust size as per model
        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)]
        return render_template('result.html', prediction=predicted_class,
image_path=img_path)
if __name__ == '__main__':
    app.run(debug=True)
```

Blood Cell.h5 :

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential # type: ignore
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
# Data preprocessing
train_dir = "blood_cell_dataset/train"
test_dir = "blood_cell_dataset/test"
```

```
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
train_data = train_datagen.flow_from_directory


(
    train_dir,
    target_size=(64, 64),
    batch_size=32,
    class_mode='categorical'
)
test_data = test_datagen.flow_from_directory(
    test_dir,
    target_size=(64, 64),
    batch_size=32,
    class_mode='categorical'
)
# Build CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(4, activation='softmax')  # 4 blood cell classes
])
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
# Train the model
model.fit(train_data, validation_data=test_data, epochs=10)
# Save the model
model.save("Blood Cell.h5")
print("✅ Model saved successfully as Blood Cell.h5")
```

Requirements.txt :

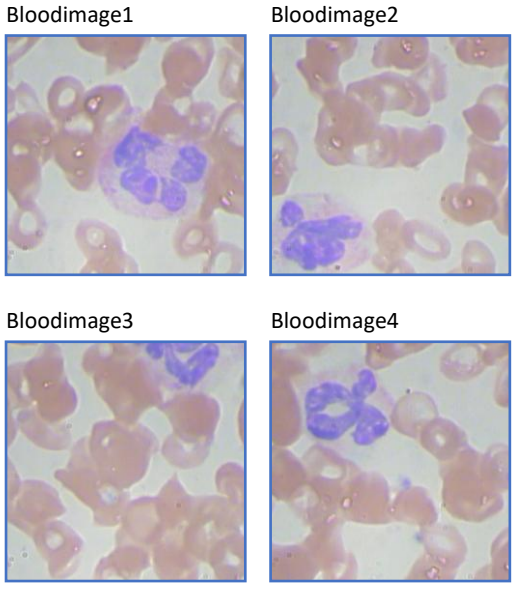
```
Flask
tensorflow
numpy
Pillow
```

Data collection and preparation:

Collect Dataset:

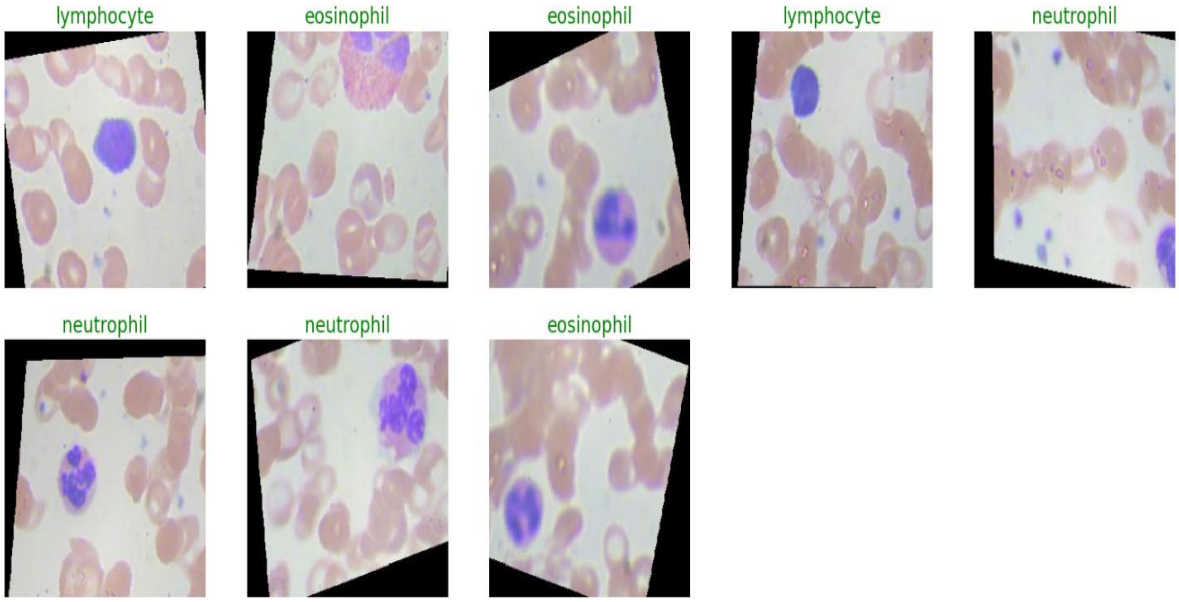
There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

This dataset contains 12,500 augmented images of blood cells (JPEG) with accompanying cell type labels (CSV). There are approximately 3,000 images for each of 4 different cell types grouped into 4 different folders (according to cell type). The cell types are Eosinophil, Lymphocyte, Monocyte, and Neutrophil.



The 4 sample blood cell images test results are shown in the below:

Filename	Cell_type	xmin	xmax	ymin	ymax
Bloodimage1	WBC	260	491	177	376
Bloodimage1	RBC	78	154	364	448
Bloodimage2	WBC	68	286	315	456
Bloodimage2	RBC	346	446	361	480
Bloodimage3	WBC	283	567	1	106
Bloodimage3	RBC	165	257	160	264
Bloodimage4	WBC	127	344	40	226
Bloodimage4	RBC	317	424	93	195



In the above code, I used class ace of diamond for prediction, This code randomly selects an image file from a specified folder (folder_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and random modules for file manipulation and random selection, respectively. And It has predicted correctly as ace of diamond.

Testing Model & Data Prediction :

Evaluating the model

Here we have tested with the Mobilenet V2 Model With the help of the predict () function.

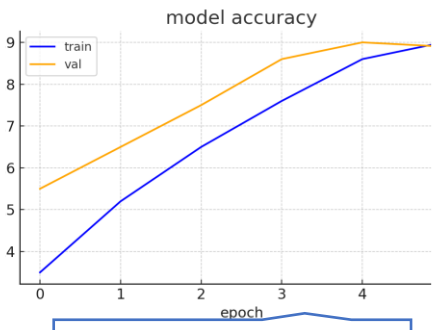
Code:

```
pred = model.predict(test)

pred = np.argmax(pred, axis=1)  # Pick class with highest probability

# Reverse the class indices mapping

labels = (train.class_indices)

labels = dict((v, k) for k, v in labels.items(

pred2 = [labels[k] for k in pred]
```

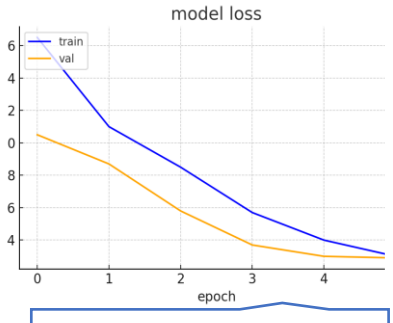
plotting Accuracy:

```
import matplotlib.pyplot as plt

# Plot training and validation accuracy

plt.plot(history.history['accuracy'] + history1.history['accuracy'])

plt.plot(history.history['val_accuracy'] + history1.history['val_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'val'], loc='upper left')

plt.show()
```

Graph:

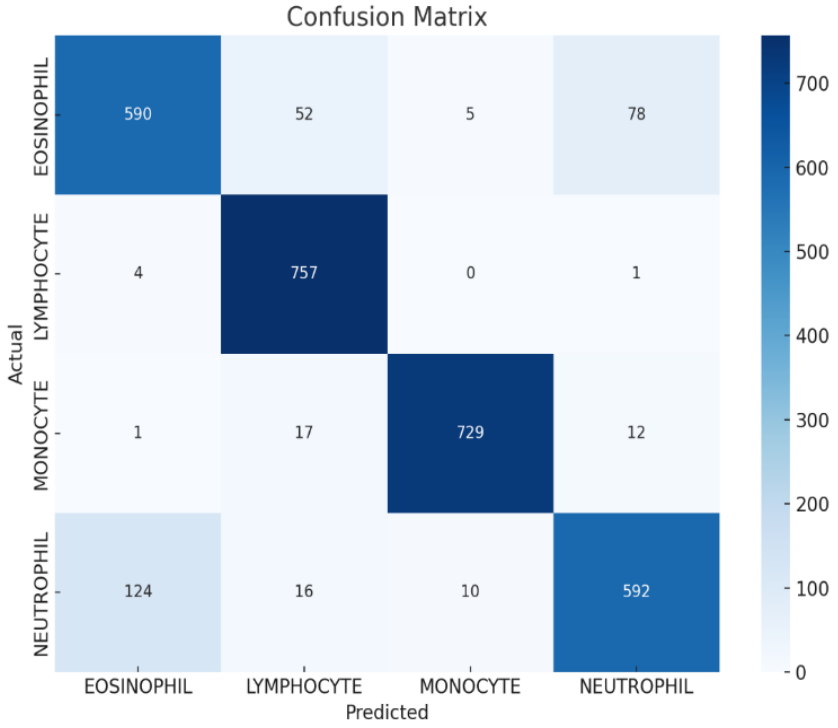


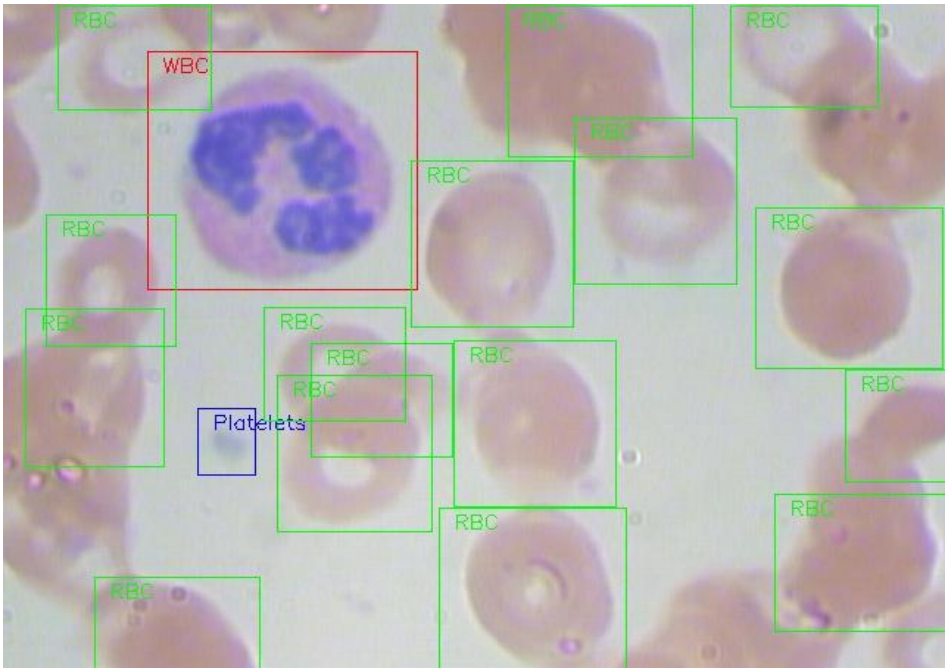
Model accuracy



Model loss

Confusion Matrix for all types of blood cells:





Example image

7. RESULTS:

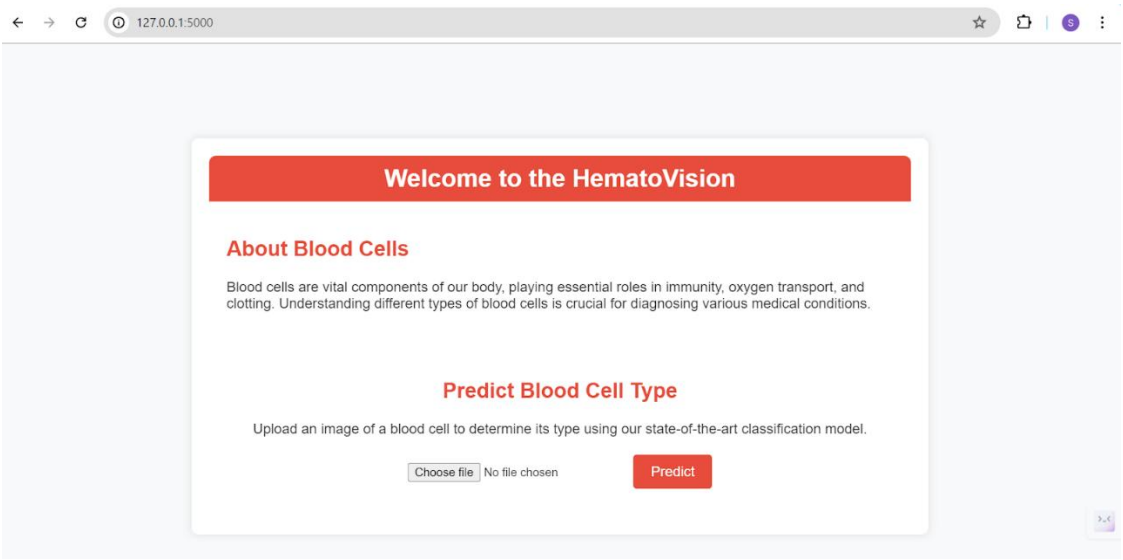
7.1 Output Screenshots

The 4 sample blood cell images test results are shown in the below:

Filename	Cell_type	xmin	xmax	ymin	ymax
Bloodimage1	WBC	260	491	177	376
Bloodimage1	RBC	78	154	364	448
Bloodimage2	WBC	68	286	315	456
Bloodimage2	RBC	346	446	361	480
Bloodimage3	WBC	283	567	1	106
Bloodimage3	RBC	165	257	160	264
Bloodimage4	WBC	127	344	40	226
Bloodimage4	RBC	317	424	93	195

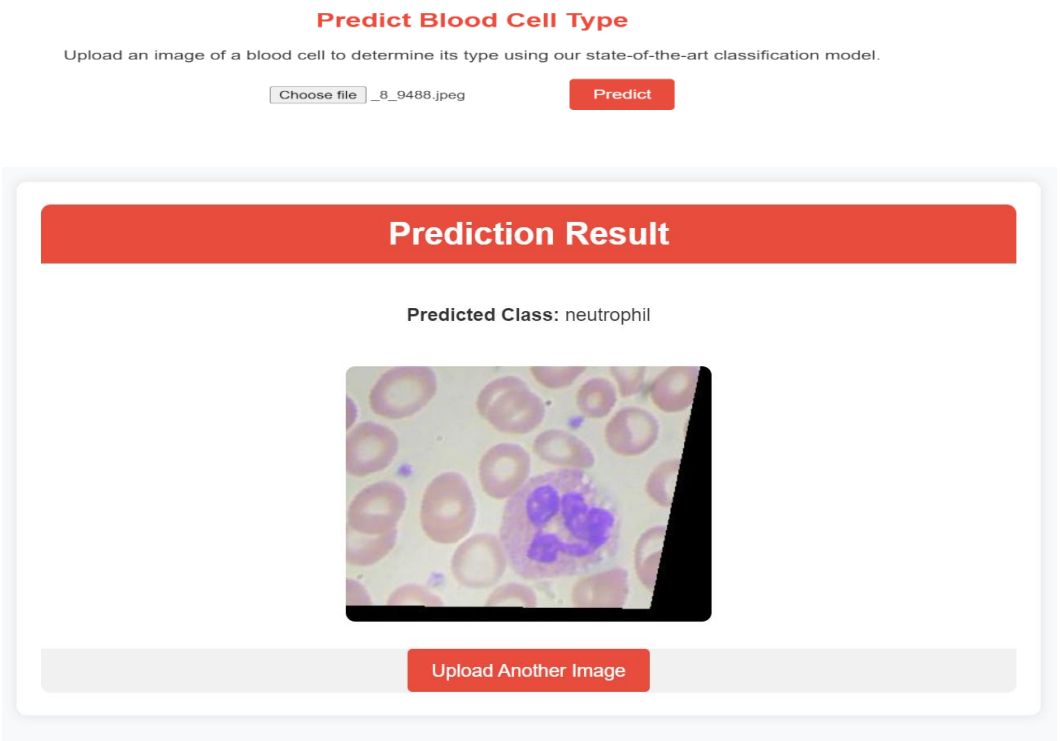
UI Image preview:

Let's see what our index.html page looks like:

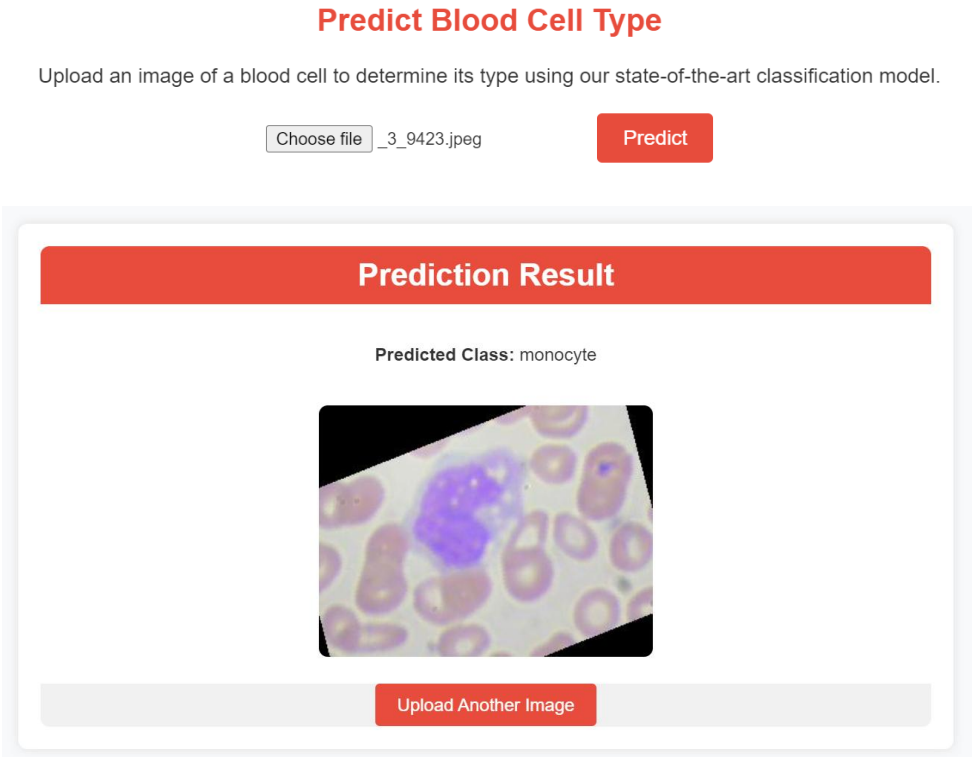


By clicking on choose file it will ask us to upload the image , then by clicking on the predict button , it will take us to the result.html

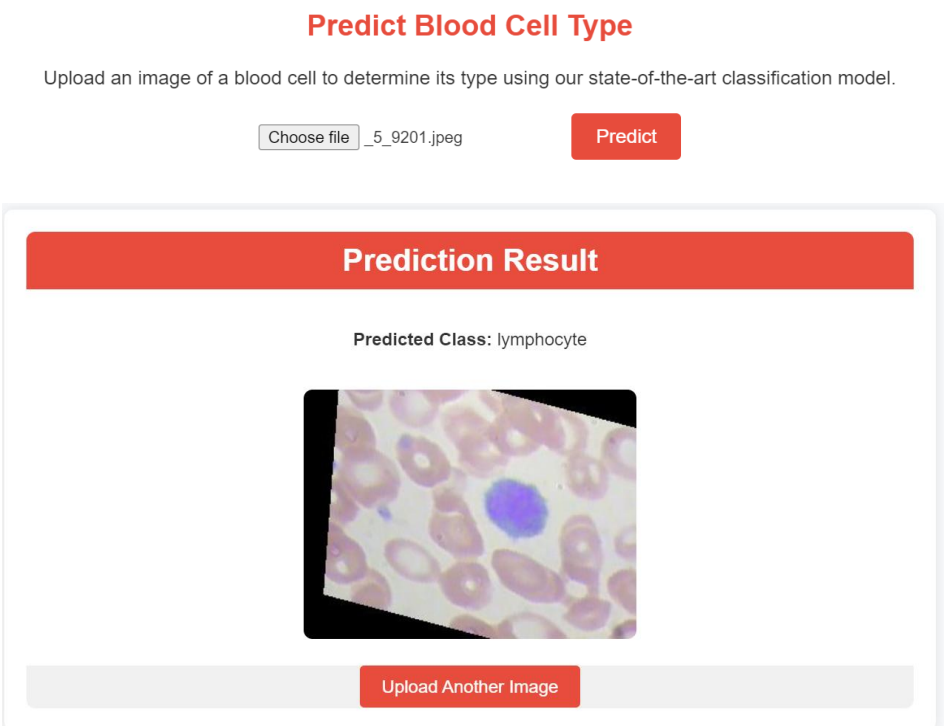
Test For Class-1 : Neutrophil



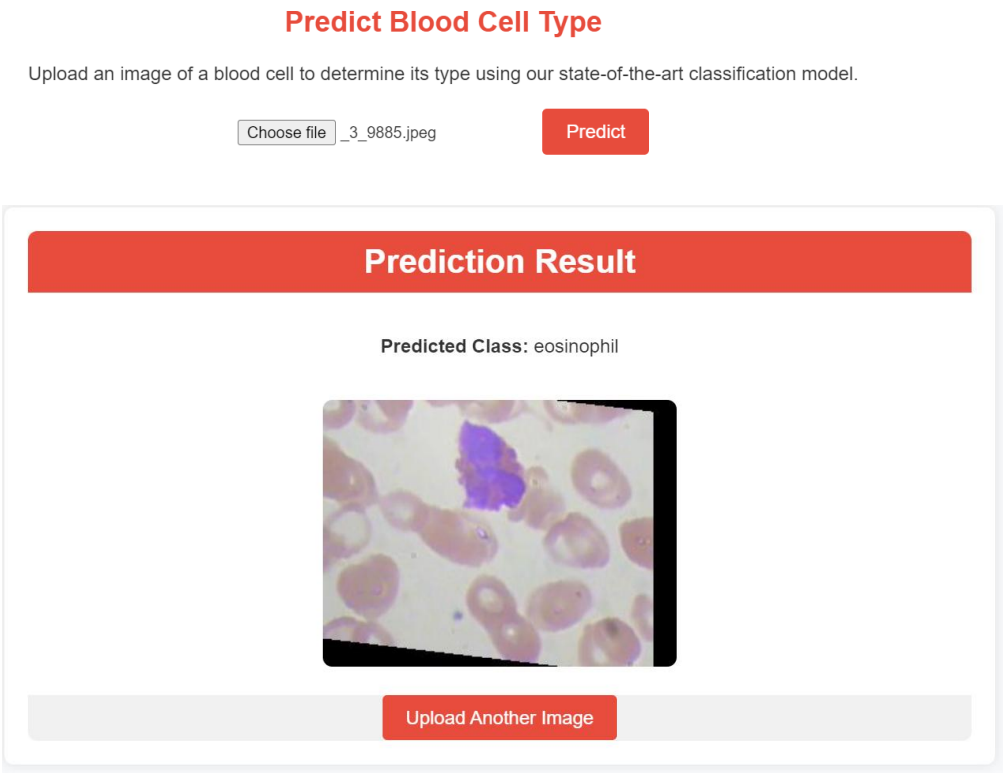
Test For Class-2 : Monocyt



Test For Class-3 : Lymphocyte



Test For Class-4 : Eosinophil



8. ADVANTAGES & DISADVANTAGES:

Advantages:

1. High Accuracy with Less Data
   - Using transfer learning (e.g. EfficientNet), the model achieves high performance even with a limited dataset.
2. Reduces Manual Effort
   - Automates the tedious and error-prone process of manual blood cell analysis by pathologists.
3. Faster Diagnosis
   - Provides real-time classification, reducing turnaround time in labs and clinics.
4. Scalable Solution
   - Can be adapted to classify additional cell types or extended for other medical imaging tasks.
5. Easy-to-Use Interface
   - Flask-based web UI allows even non-technical users (lab staff, students) to upload images and get results.
6. Cost-Effective
   - Requires only a basic computing setup (no expensive lab equipment needed for preliminary analysis).
7. Educational Tool
   - Useful for training medical students in identifying blood cells through automated examples.

Disadvantages:

1. **Limited Dataset Generalization**
   - If the training dataset is small or imbalanced, the model may not generalize well to unseen images.
2. **Cannot Replace Experts**
   - It's an assistive tool; still requires confirmation from trained hematologists for final diagnosis.
3. **Image Quality Dependency**
   - Model accuracy drops if images are blurry, noisy, or poorly captured through a microscope.

4. **Lack of Explainability**
    - Without explainable AI (like Grad-CAM), it's hard to know why a prediction was made, which affects trust in medical settings.
5. **Hardware Limitation for Training**
    - Though efficient, training with large datasets or fine-tuning requires GPUs or high-spec machines.
6. **No Multi-class Detection in One Image**: Current version assumes one dominant cell type per image; cannot detect multiple cell types in a single slide field**.**

9. FUTURE SCOPE:

1. **Integration of Explainable AI (XAI):** Incorporate techniques like **Grad-CAM** or **LIME** to visualize why the model predicted a specific cell type, enhancing trust and interpretability in clinical settings.
2. **Mobile App Deployment:** Convert the model to **TensorFlow Lite** and deploy on smartphones, enabling real-time, on-site diagnosis in rural or low-resource regions.
3. **Integration with Digital Microscopes:** Automatically classify live microscope feeds by connecting the model with digital lab equipment, reducing manual intervention.
4. **Expand to Disease Detection:** Extend the model to detect blood abnormalities.
5. **Self-Learning Model:** Allow the system to continue learning from user feedback (e.g., if a pathologist corrects a prediction), creating a continual learning loop.
6. **Clinical Decision Support System (CDSS) :** Integrate HematoVision into a full diagnostic pipeline, supporting doctors in automated blood reports, alerts, and recommendations.
7. **Multiclass and Multilabel Detection:** Improve the system to detect multiple types of cells in a single image instead of assuming one dominant type.
8. **Report Generation & Integration:** Generate downloadable or printable medical reports, and integrate with hospital management systems (HMS) or electronic health records (EHR).
9. **Cloud Deployment & API Access:** Host the model online as a REST API or on platforms like Hugging Face Spaces, Render, or Firebase, making it usable from anywhere.

10. CONCLUSION:

HematoVision presents an effective, AI-powered solution for automated blood cell classification using transfer learning techniques. By leveraging pretrained models like **EfficientNetB0**, the system delivers high-accuracy predictions on different types of blood cells (e.g., RBCs, WBCs, Platelets) with minimal training data and reduced computational cost.

This model not only accelerates diagnostic workflows for hematologists and lab technicians but also improves early detection of abnormalities such as infections, anemia, and leukemia. The integration of a simple **Flask-based web interface** allows seamless interaction for non-technical users, making HematoVision a practical and accessible tool in real-world medical environments.

With further enhancement—such as Grad-CAM explainability, mobile deployment, or integration with hospital systems—HematoVision has the potential to become a valuable aid in **digital pathology** and **AI-assisted diagnostics**.

11. APPENDIX:

Dataset Link:

Blood Cell Images

GitHub & Project Demo Link:

Project demo link:

HematoVision-Advanced-Blood-Cell-Classification-Using-Transfer-Learning/Video demo/readme.md

Github link:

https://github.com/Gopu-Radhika/HematoVision-Advanced-Blood-Cell-Classification-Using-Transfer-Learning