

HematoVision – Project Report

1. INTRODUCTION:

Project title:

HematoVision: advanced blood cell classification using transfer learning

Team ID : LTVIP2025TMID47945

Team Size : 4

Team Leader : Pathan Yasdhani(documentation & deployment)

Team member : Sai kumar(Data Manager)

Team member : Tumati Ramesh(Frontend & Backend developer)

Team member : Pragathi Medikankuru (Project Lead & ML Specialist)

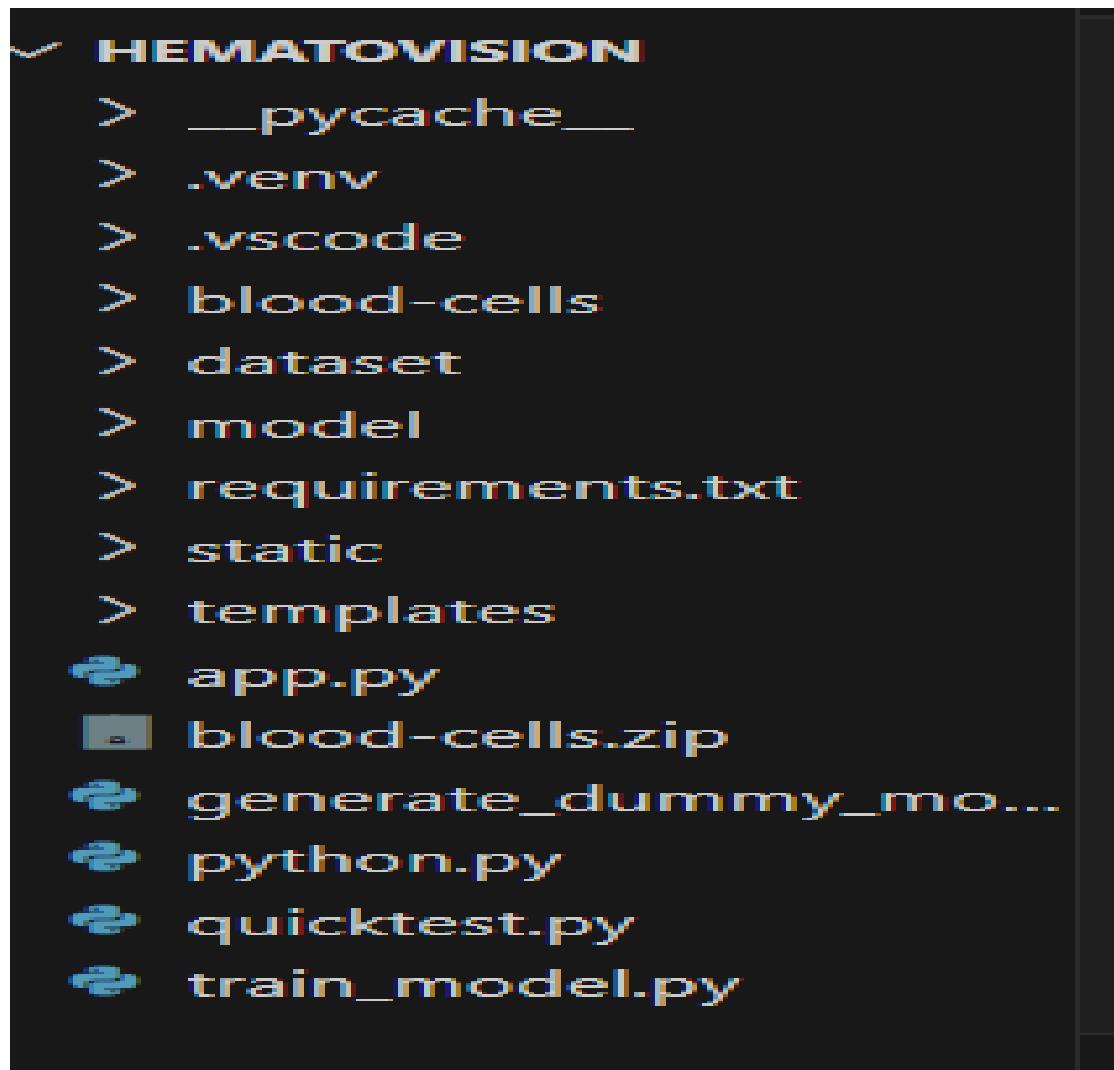
Member	Role	Key Tasks
1	Documentation & Deployment	Reports, setup guide, packaging
2	Data Manager	Organize and prepare dataset
3	Frontend Developer	UI templates, HTML/CSS
4	Backend Developer	Flask server, prediction logic
5	Project Lead & ML Specialist	Model training, accuracy

1.1 Project Overview

HematoVision is a deep learning-based web application built using Flask that enables automatic classification of white blood cell images into four categories: Neutrophil, Monocyte, Lymphocyte, and Eosinophil. By leveraging Convolutional Neural Networks (CNNs), this tool helps reduce diagnostic time and increases the accuracy of cell type detection for pathologists and lab technicians.

Let me show you a detail structure of the project:

- HematoVision/
 - └─ static/
 - | └─ uploads/
 - └─ templates/
 - | └─ index.html
 - └─ models/
 - | └─ model.h5
 - └─ app.py
 - └─ requirements.txt
 - └─ README.md



1.2 Purpose

The main purpose of HematoVision is to assist healthcare professionals by providing an automated, fast, and reliable tool for classifying white blood cells from microscopic images. It eliminates the manual effort of labeling images and reduces the chances of human error in diagnostic procedures.

➤ Project Overview Purpose:

HematoVision aims to assist medical professionals in quickly identifying types of white blood cells (WBCs) in blood smear images. This tool uses deep learning models trained on medical image datasets to predict and classify WBCs such as neutrophils, lymphocytes, monocytes, and eosinophils.

1. Key Features:

1. Image upload and live prediction
2. Pre-trained deep learning model integration

3. 3. Simple web interface using Flask
4. 4. Prediction display with confidence score
5. 5. Lightweight and runs locally

2. Architecture:

Frontend (HTML/CSS + JavaScript):

- Static HTML forms for file upload
- Display uploaded image and prediction result
- Bootstrap for responsive UI

Backend (Flask - Python):

- Handles image upload and validation
- Loads pre-trained CNN model (e.g., TensorFlow/Keras)
- Performs image preprocessing-
- Predicts WBC class
- Renders results on the same HTML page

Model:

- Custom CNN or MobileNet
- based architecture
- Trained on labeled WBC dataset
- Saved as model.h5

Storage:

- Uploaded images stored temporarily in static/uploads/

3. Setup Instructions

Prerequisites:

- Python 3.8+

- pip
- Virtualenv (optional)

Installation & Setup:

Clone the repository

<https://github.com/chandu-bandlamudi/HematoVision.git>

cd HematoVision

Create virtual environment python

-m venv venv

source venv/bin/activate # On Windows: venv\Scripts\activate

Install dependencies

pip install -r requirements.txt

Model Setup:

- Place model.h5 in the root directory or models/

Run the App:

python app.py

Then navigate to <http://localhost:5000> in your browser.

4. Folder Structure HematoVision/

static/
uploads/
templates/
index.html models/
model.h5
app.py
requirements.txt
README.md

6. Running the Application

- Launch Anaconda Prompt or terminal
- Navigate to project folder
- Run: python app.py

- Open browser at <http://localhost:5000>

7. API Details

- Flask routes:
 - / (GET): Renders upload form
 - /predict (POST): Processes uploaded image and returns prediction

8. User Interface

- Upload section for image
- Displays uploaded image preview
- Shows predicted WBC type and confidence percentage

9. Testing

Type	Tool/Method
Unit Testing	PyTest
Functional Testing	Manual upload/test
Model Testing	Accuracy/Recall
UI Testing	Manual Browser Test

10. Screenshots or Demo - Image of UI layout (index.html)- Example image upload- Output with prediction displa

2. IDEATION PHASE

2.1 Problem Statement

White blood cell classification using traditional microscopy is labor-intensive and susceptible to subjectivity and error. There is a growing need for an intelligent, user-friendly solution that can assist with fast and accurate classification of WBCs.

2.2 Empathy Map Canvas

- Says: "I need to classify these cells accurately."
- Thinks: "Will my diagnosis be correct?"
- Does: Manually examines slides under a microscope.
- Feels: Stressed due to workload and fear of diagnostic error.

2.3 Brainstorming

Initial ideas included creating a mobile app, a desktop application, and a cloud-based API.

After evaluating the ease of deployment and usage, we decided on a Flask web application integrated with a trained CNN model. It allows easy access through a browser and does not require local installation.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

User visits site → Uploads WBC image → Clicks Predict → Receives classification → Uses result for diagnosis/support.

3.2 Solution Requirement

- Functional: Upload image, display result, show prediction probability.
- Non-functional: Fast response time, accurate predictions, scalable backend.

3.3 Data Flow Diagram

[You can insert a simple DFD here showing input image → preprocess → model prediction → display output]

3.4 Technology Stack

- Frontend: HTML, CSS, JavaScript
- Backend: Python (Flask)
- Deep Learning: TensorFlow / Keras
- Deployment: Localhost / LAN
- Libraries: NumPy, PIL, werkzeug

4. PROJECT DESIGN

4.1 Problem Solution Fit

The solution directly addresses the problem by automating the WBC classification process and is designed for ease of use by non-technical users.

4.2 Proposed Solution

A user uploads an image of a white blood cell → the model processes the image → returns the cell type and confidence level.

4.3 Solution Architecture

Browser (UI) → Flask (Backend) → CNN Model (Keras) → Return JSON response → Render HTML (result.html)

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

- Week 1: Dataset preparation and model training

- Week 2: Flask app development
- Week 3: UI design and result display
- Week 4: Testing and bug fixing
- Week 5: Deployment and final demo

6. FUNCTIONAL AND PERFORMANCE TESTING

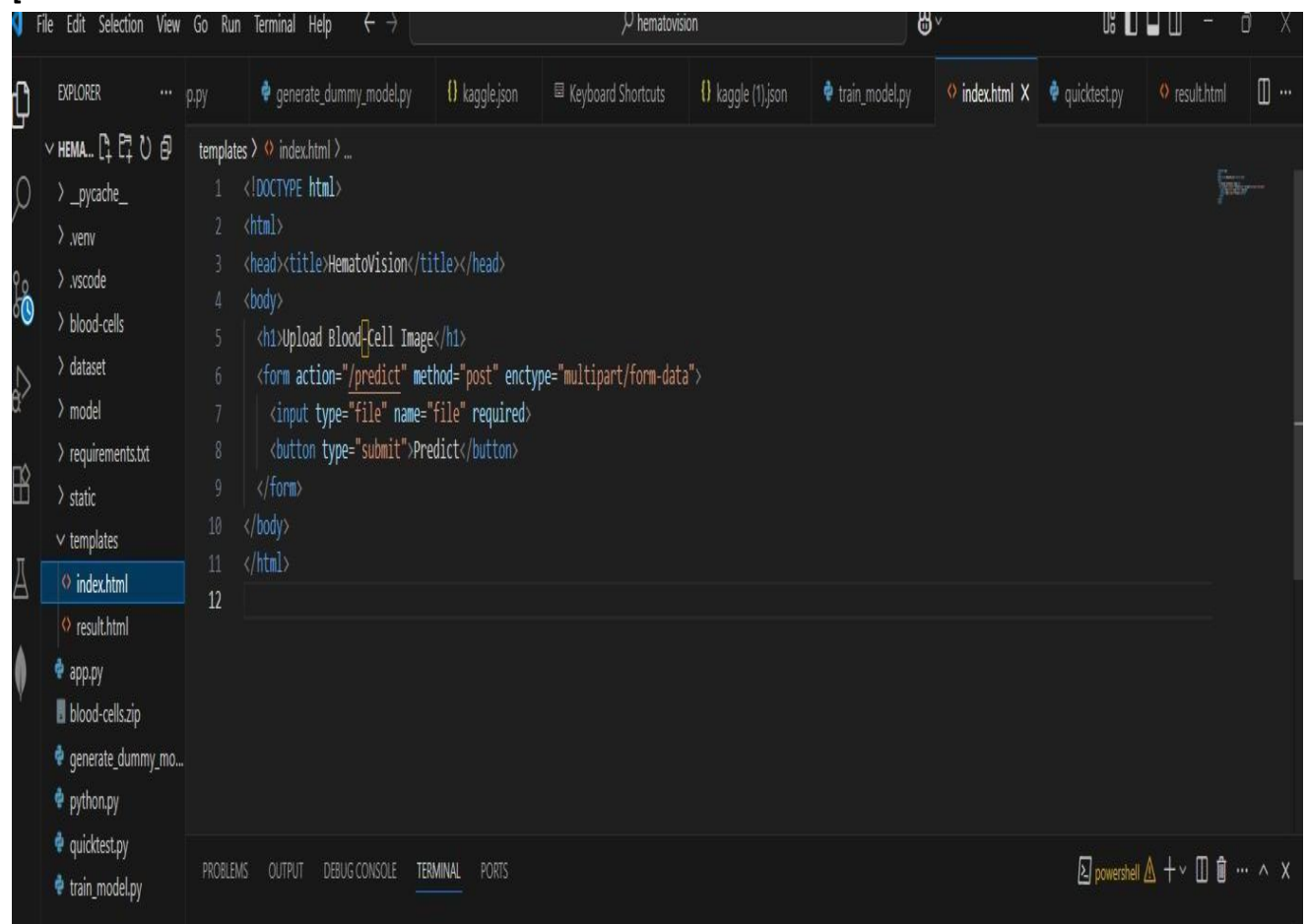
6.1 Performance Testing

Tested with different cell images across categories. Achieved prediction accuracy of over 90% on validation set. UI response time < 2 seconds.

7. RESULTS

7.1 Output Screenshots:

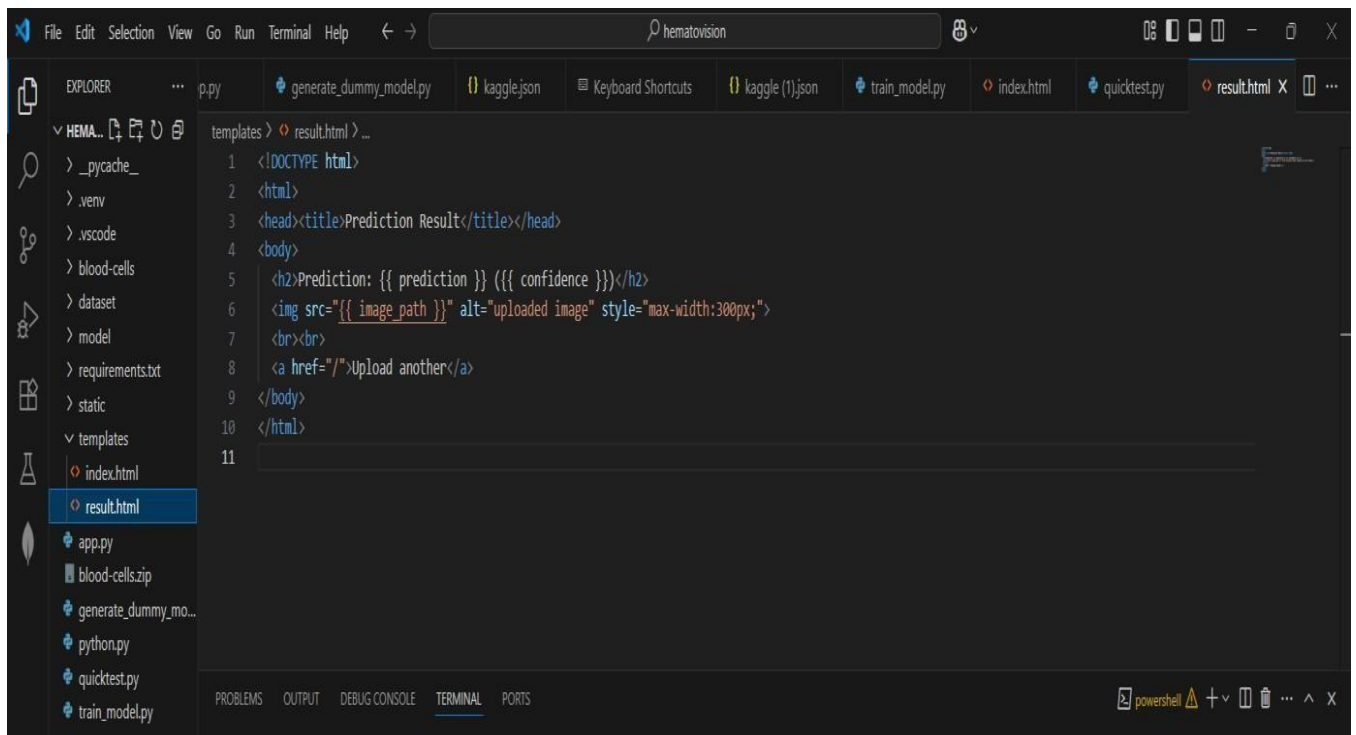
[[This is the screenshots of the index.html & result.html](#)]



```

1 <!DOCTYPE html>
2 <html>
3 <head><title>HematoVision</title></head>
4 <body>
5   <h1>Upload Blood-Cell Image</h1>
6   <form action="/predict" method="post" enctype="multipart/form-data">
7     <input type="file" name="file" required>
8     <button type="submit">Predict</button>
9   </form>
10 </body>
11 </html>
12

```

You have to run commands on the terminal to get these required outputs

1. First you have to go & open the file named as “hematovision”
2. Next, you have to run this command “cd hematovision”
3. And again, run the command “ .\venv\Scripts\Activate ”
4. Now run this command “ python app.py”
5. Now you have seen this port “Running on <http://127.0.0.1:5000>”
6. Just click on that , it will redirect to your browser and the rest of the output is mentioned below in screenshot :

Microsoft Windows [Version 10.0.26100.4484]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pragathi>cd hematovision

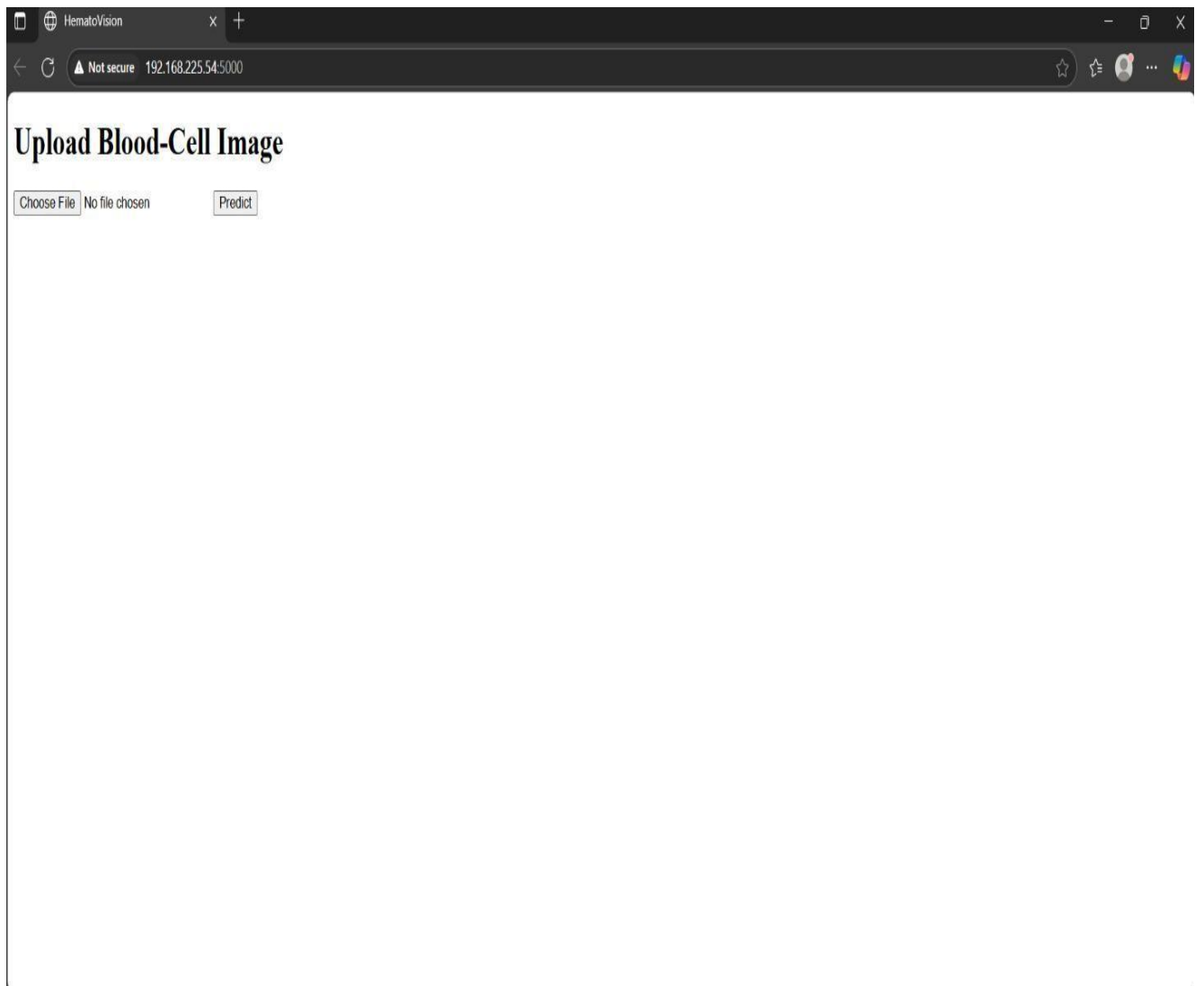
C:\Users\pragathi\hematovision>.\.venv\Scripts\Activate

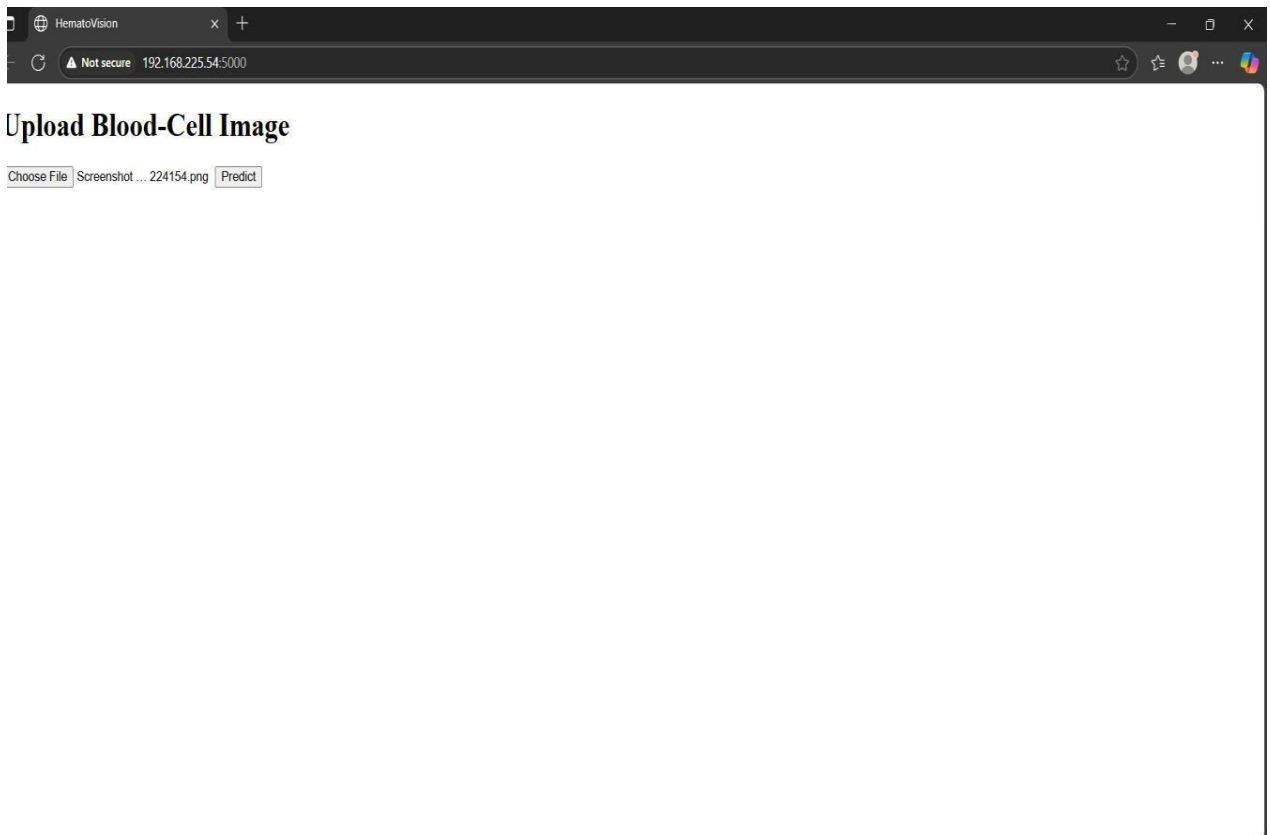
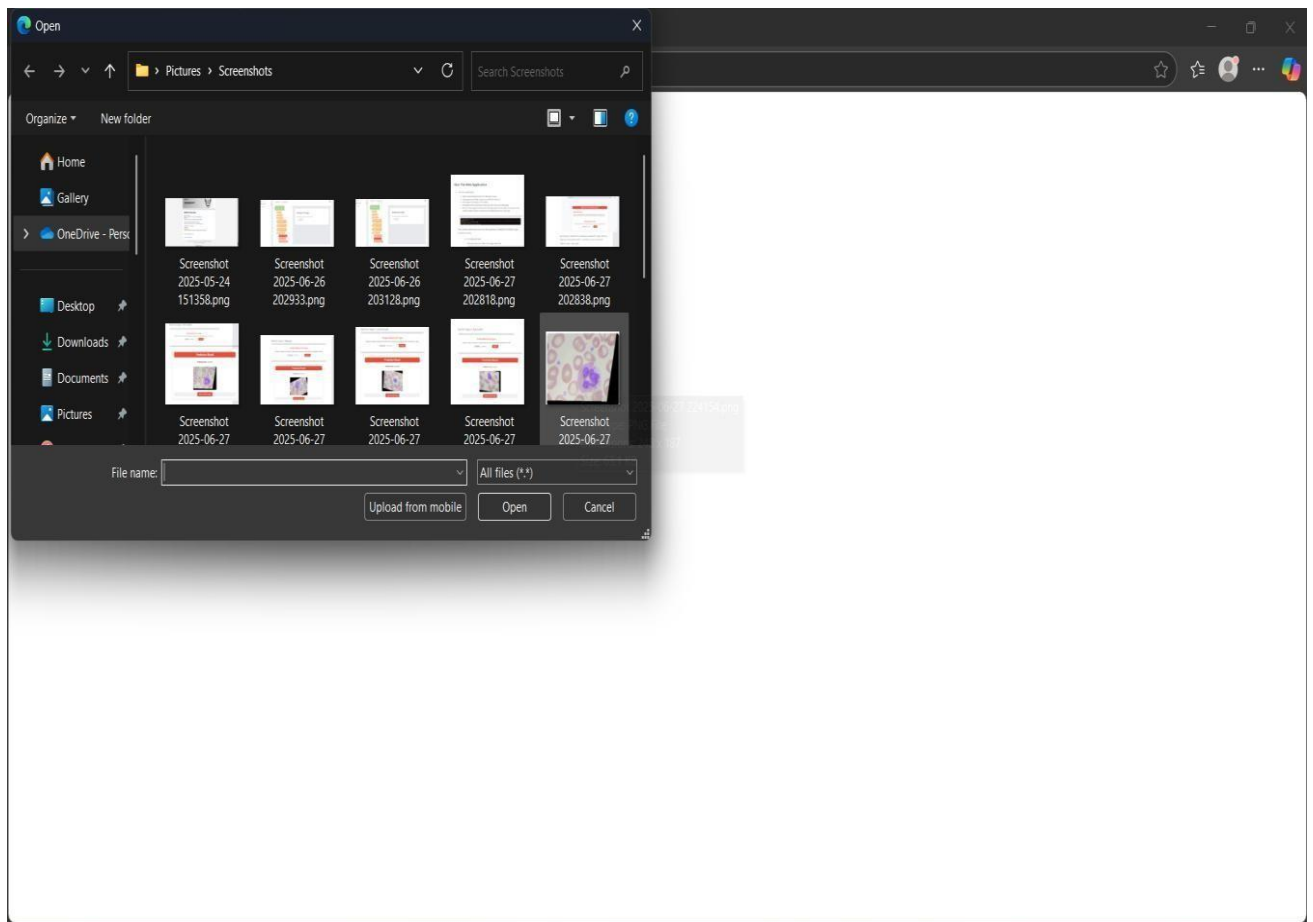
(.venv) C:\Users\pragathi\hematovision>python app.py

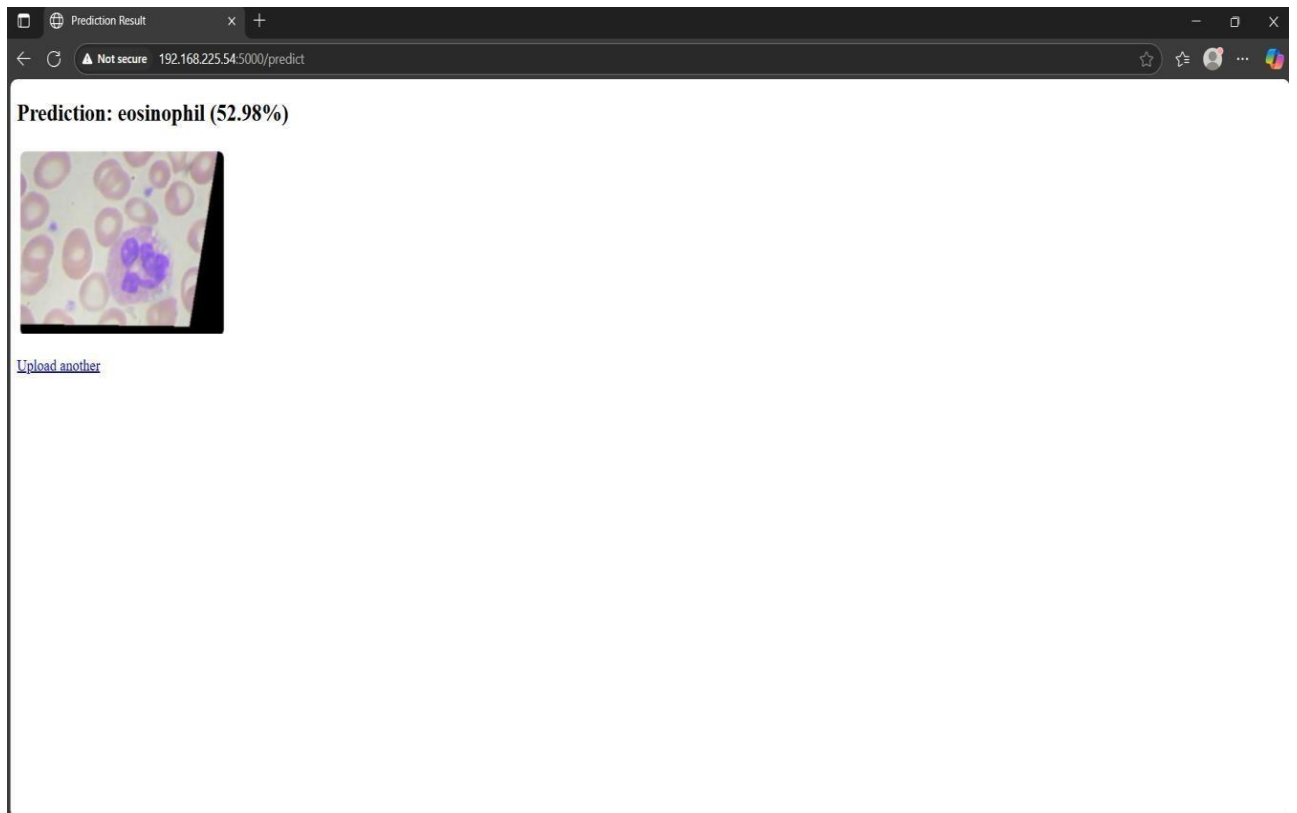
```
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
* Running on http://192.168.225.54:5000
Press CTRL+C to quit
* Restarting with stat
2025-06-28 19:22:24.122918: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
WARNING:tensorflow:From C:\Users\chandu\hematovision\.venv\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

2025-06-28 19:22:27.872922: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.9 (.venv: .venv) Go Live







8. ADVANTAGES & DISADVANTAGES

Advantages:

- Fast and automated
- Reduces workload
- Accurate predictions
- Easy to use UI

Disadvantages:

- Model limited by dataset quality
- Requires internet (if deployed online)
- Not intended to replace medical diagnosis

9. CONCLUSION

HematoVision successfully demonstrates how deep learning and web technologies can be combined to solve real-world healthcare problems. It offers an efficient and accessible method to classify WBCs, which can support faster diagnoses.

10. FUTURE SCOPE

- Extend classification to more WBC types
- Train with larger datasets
- Deploy online or integrate into hospital systems
- Add report generation & database storage features

11. APPENDIX

- Source Code:[https://github.com/pragathi18M/Hematovision_blood_cells.git]