### HematoVision: Advanced Blood Cell Classification Using Transfer Learning

Category: Artificial Intelligence

Skills Required:  
Python,Deep Learning,Transfer Learning

Project Description:

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.

Scenario 1:Automated Diagnostic Systems for Healthcare

Integrating HematoVision into automated diagnostic systems in clinical settings can revolutionize blood analysis. By using transfer learning, the system quickly adapts to the specifics of blood cell classification, capturing images of blood samples, classifying the cells in real-time, and generating detailed reports. This automation reduces the manual workload on pathologists, speeds up diagnostic processes, and ensures high accuracy in results, ultimately improving patient care and treatment efficiency.

Scenario 2: Remote Medical Consultations

HematoVision can be employed in telemedicine platforms to enhance remote consultations and diagnostics. With transfer learning, the model's ability to accurately classify blood cells from diverse sources is improved, allowing healthcare providers to upload blood cell images for automated analysis. This enables timely and accurate assessments without the need for in-person visits, facilitating better access to specialized medical expertise and improving healthcare delivery in remote or underserved areas.

### HematoVision's transfer learning-based classification model can be integrated into educational tools for medical training. By incorporating this advanced technology into interactive learning platforms, students and laboratory technicians can upload and analyze blood cell images to receive instant feedback. This hands-on learning experience enhances their understanding of blood cell morphology and classification, providing practical skills and knowledge that are crucial for accurate diagnostic practice and medical training and. Project Objectives

By the end of this project, you will:

* Know fundamental concepts and techniques used for Deep Learning.
* Gain a broad understanding of data.
* Have knowledge of pre-processing the data/transformation techniques on outliers and some visualization concepts.

**Project Flow**

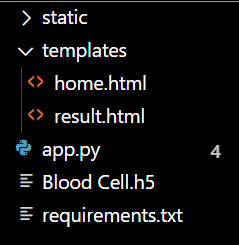
* The user interacts with the UI (User Interface) to choose the image.
* The chosen image is analyzed by the model which is integrated with the flask application.
* Once the model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

* Data Collection: Collect or download the dataset that you want to train.
* Data pre-processing
  + Data Augmentation
  + Splitting data into train and test
* Model building
  + Import the model-building libraries
  + Initializing the model

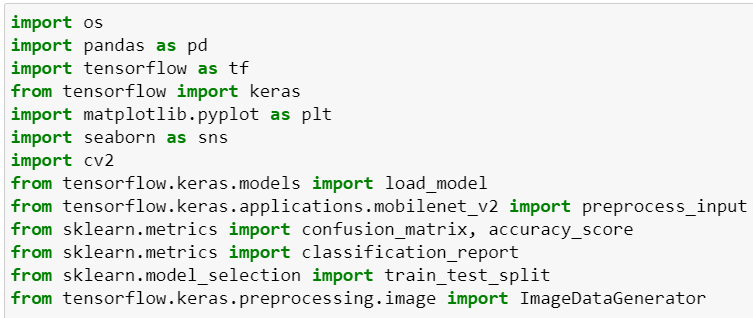
### Project Structure

Create the Project folder which contains files as shown below



* We are building a Flask application with HTML pages stored in the templates folder and a Python script app.py for scripting.
* Blood Cell.h5 is our saved model. Further, we will use this model for flask integration.

Import the necessary libraries as shown in the image.



Activity 1.2: Read the Dataset:

* Our dataset format might be in .csv, excel files, .txt, .json, or zip files, etc. We can read the dataset with the help of pandas.

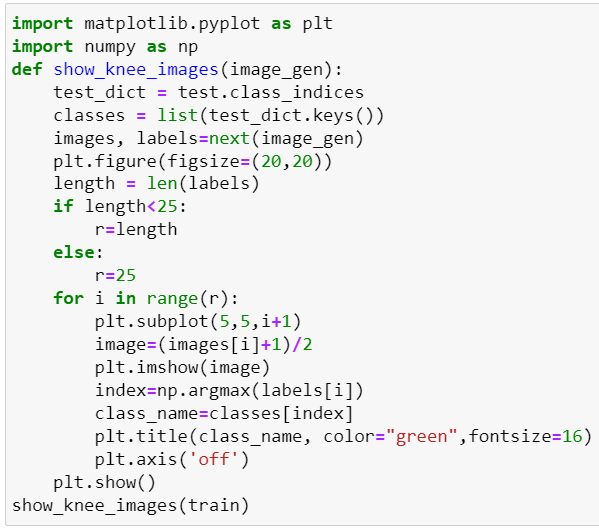
At first, unzip the data and convert it into a pandas data frame.

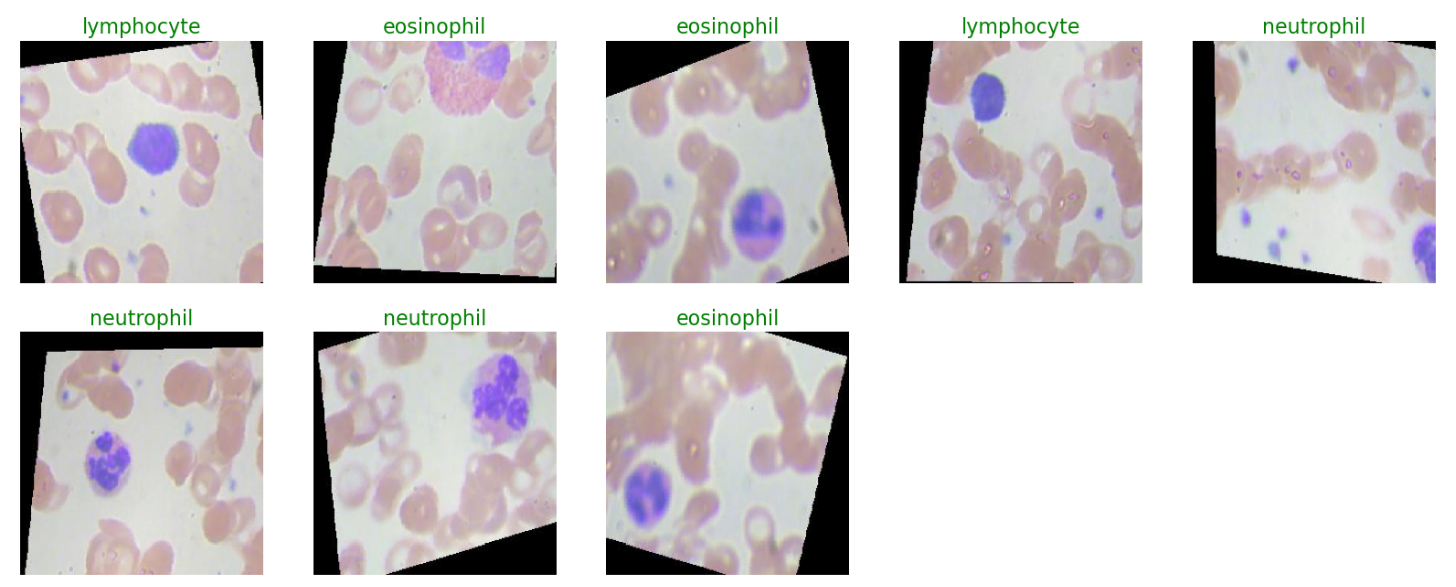


* + Training and testing the model
  + Evaluating the performance of the model
  + Save the model
* Application Building
  + Create an HTML file
  + Build python code

### Data Visualization

The provided Python code imports necessary libraries and modules for image manipulation. It selects a random image file from a specified folder path. Then, it displays the randomly selected image using IPython's Image module. This code is useful for showcasing random images from a directory for various purposes like data exploration or testing image processing algorithms.



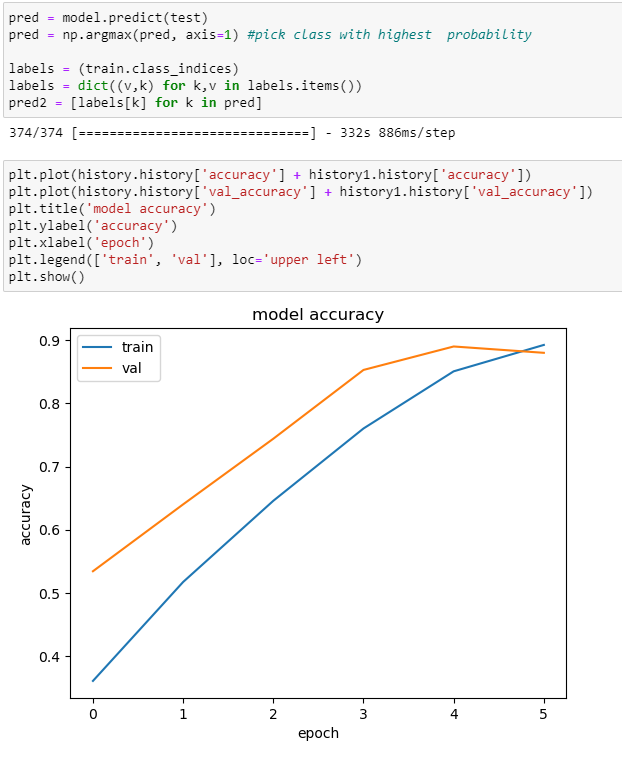


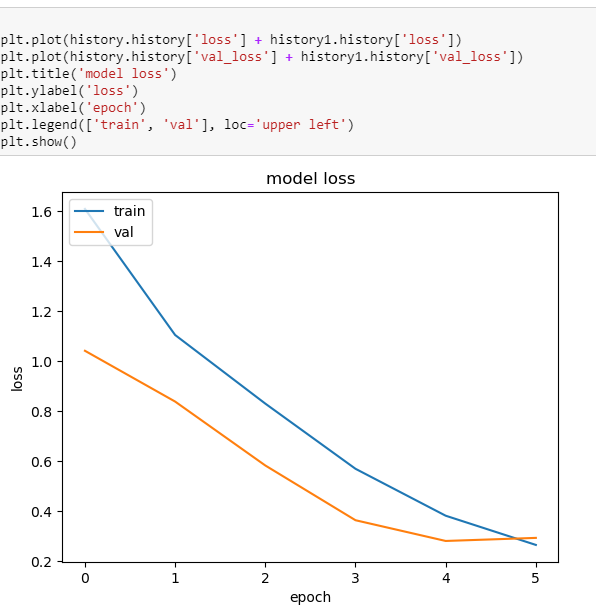
In the above code, I used class ace of diamond  for prediction, This code randomly selects an image file from a specified folder (folder\_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and random modules for file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

### Testing Model & Data Prediction

Evaluating the model

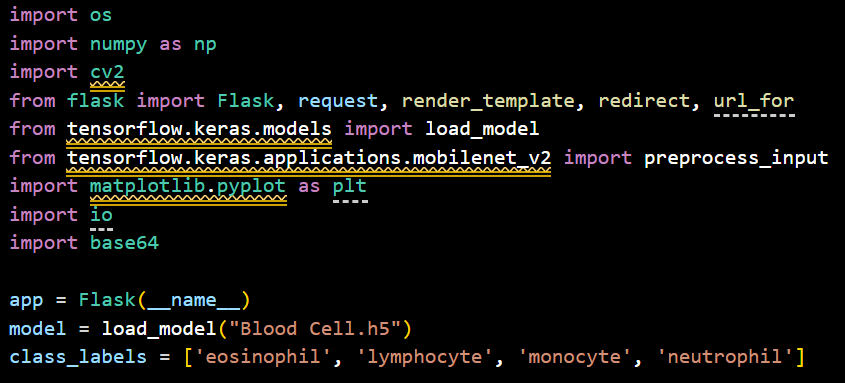
Here we have tested with the Mobilenet V2 Model With the help of the predict () function.





### Build Python code:

Import the libraries

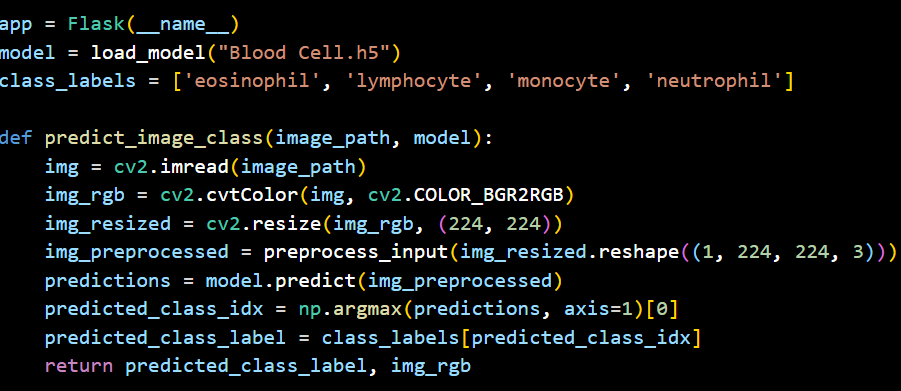


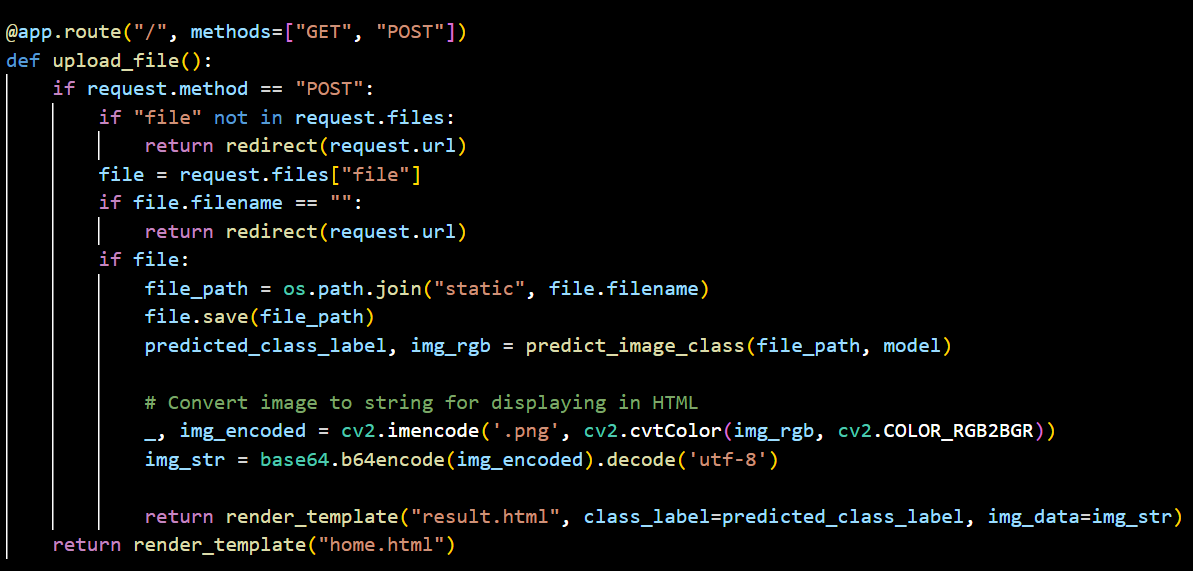
Load the saved model. Importing the Flask module in the project is mandatory. An object of the Flask class is our WSGI application. The Flask constructor takes the name of the current module (\_\_name\_\_) as argument.

Here we will be using the declared constructor to route to the HTML page which we have created earlier.

In the above example, the ‘/’ URL is bound with the index.html function. Hence, when the index page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

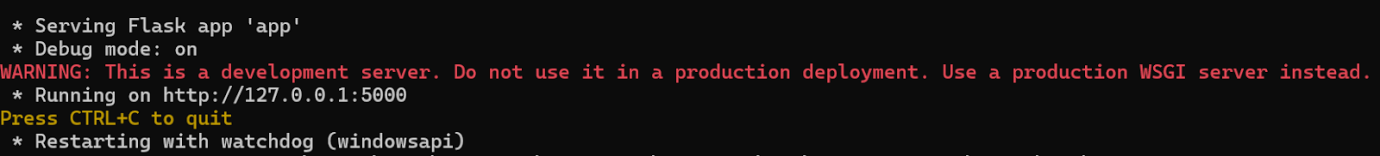




Here we are routing our app to the output() function. This function retrieves all the values from the HTML page using a Post request. That is stored in an array. This array is passed to the model. Predict () function. This function

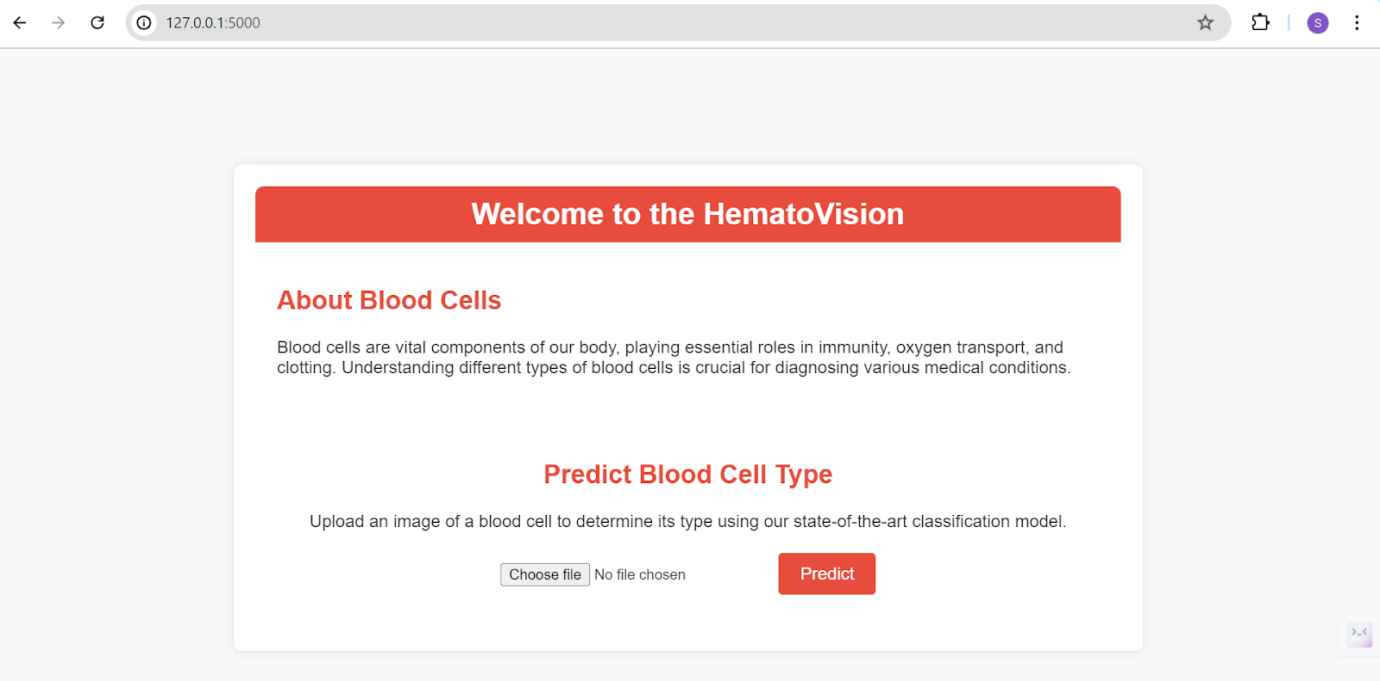
 Run the application

* Open Anaconda prompt from the start menu
* Navigate to the folder where your Python script is.
* Now type the “app.py” command
* Navigate to the local host where you can view your web page.
* Click on the inspect button from the top right corner, enter the inputs, click on the predict button, and see the result/prediction on the web.



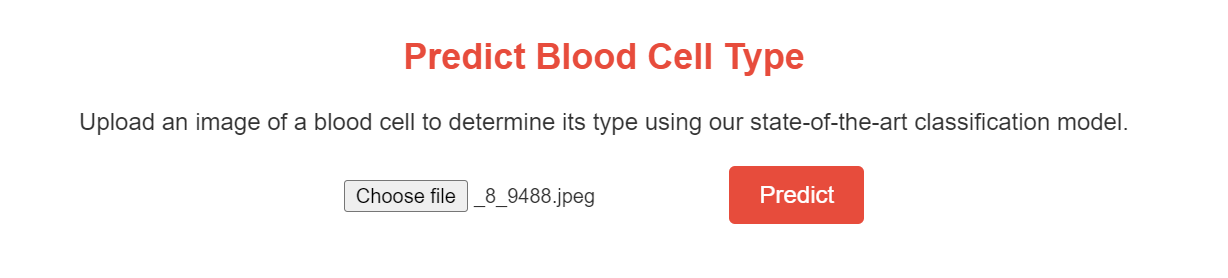
Now, Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below results

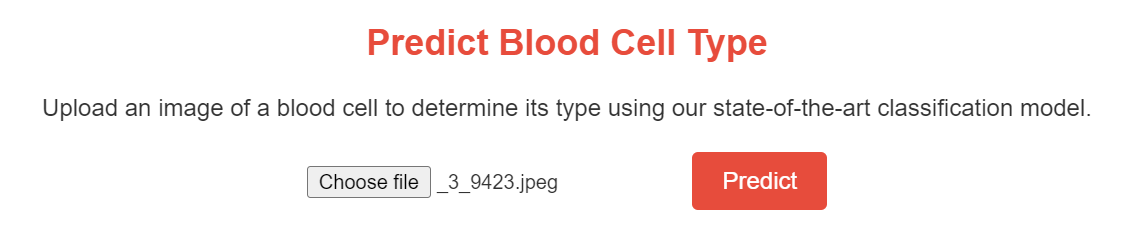
* UI Image preview:

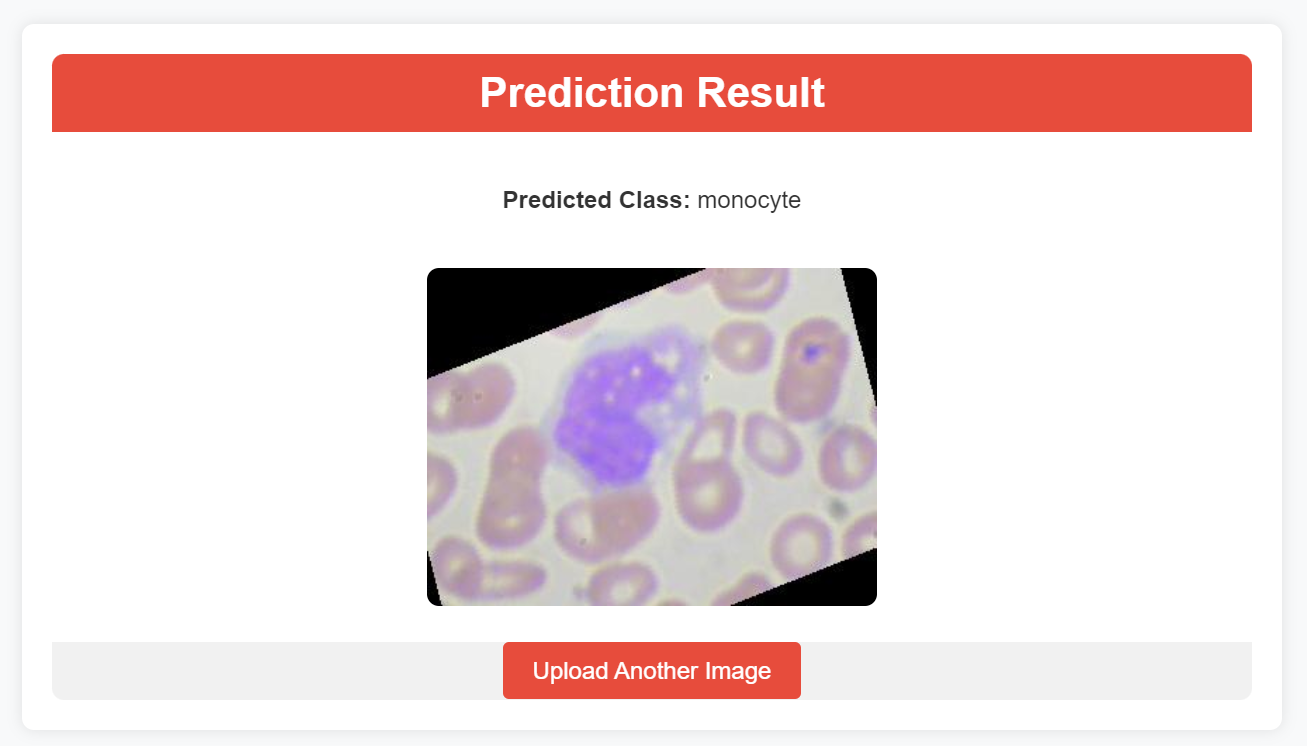
Let’s see what our index.html page looks like:

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

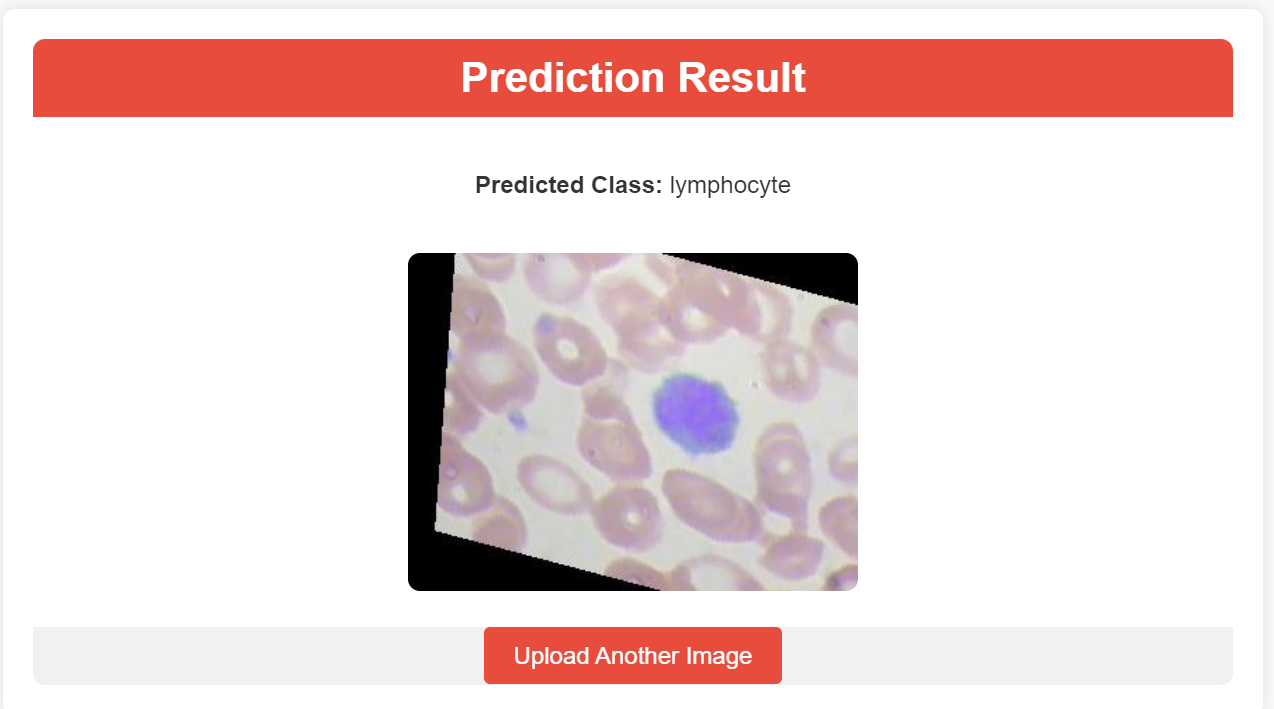
Test For Class-1 : Neutrophil



Test For Class-2 : Monocyt



* Test For Class-3 : Lymphocyte



Test For Class-4 : Eosinophil

