**Project Design Phase**

**Problem – Solution**

|  |  |
| --- | --- |
| Date | 28 june 2025 |
| Team ID | LTVIP2025TMID59640 |
| Project Name | hematovision |
| Maximum Marks | 2 Marks |

**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.

**Solution:**

1. **Model Training and Saving**A deep learning model (e.g., MobileNetV2) was trained to classify blood cell types (eosinophil, lymphocyte, monocyte, neutrophil) using a labeled image dataset. The trained model was saved as Blood Cell.h5.
2. **Flask Backend for Deployment**  
   A Flask web server (app.py) was built to load the trained model, receive image uploads, preprocess them, make predictions, and return the result.
3. **Frontend with HTML Templates**Two HTML pages (home.html for uploading and result.html for showing predictions) were placed in the templates/ folder, enabling a clean and user-friendly interface.
4. **Image Upload and Prediction**Users upload a cell image via the web UI. Flask saves the image, processes it, and feeds it into the model. The predicted class label is shown on the result page along with the uploaded image.
5. **Local Web App Execution**The app is run locally using python app.py, and accessed via <http://127.0.0.1:5000>. The user interacts via browser and gets real-time predictions.