Project Documentation Format

1. Introduction

Project Title:
HematoVision: Transfer Learning-Based Blood Cell Classification for Medical Imaging

Team Members:

- C Yateesh Kumar – UI Design & Testing
- Bhupathi Aasritha– Data Preparation
- Boreddy Likitha- Documentation
- Boya Rajesh– Backend Developer

2. Project Overview

Purpose:
To develop an AI-based system that detects and classifies blood cells using image classification via a pretrained MobileNetV2 model. The goal is to assist healthcare workers and researchers in identifying blood cell types for improved medical diagnostics.

Key Features:

- Upload blood cell images using a web interface
- Predict cell type using pretrained MobileNetV2
- Display cell name and confidence score
- Lightweight, fast and user-friendly UI
- No database used; all processing is in-memory

3. Architecture

Frontend:
- HTML and CSS (via Flask templates)

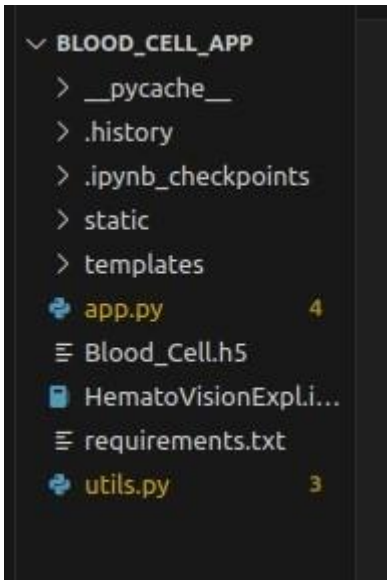
Backend:
- Flask for web framework and routing
- TensorFlow/Keras to load and run the MobileNetV2 model (Blood_Cell.h5)
- OpenCV for image processing
- Prediction logic in a separate utils.py script

Database:
- Not used; images and predictions are processed in-memory.

4. Setup Instructions

1. Install dependencies:
   pip install -r requirements.txt
2. Place the model file (Blood_Cell.h5) in the root directory.
3. Run: python app.py
4. Open your browser at http://127.0.0.1:5000/

5. Folder Structure



6. Running the Application

Run the Flask application locally with: python app.py
Then visit http://127.0.0.1:5000 in your browser.

7. API Documentation

Endpoint	Method	Description
`/`	GET	Home page (upload form)
`/upload`	POST	Accept image and trigger prediction
`/result`	GET	Show prediction result

8. Authentication

- No authentication is used.

- The application runs locally and is accessible via `http://127.0.0.1:5000`.

💻 9. User Interface

- Clean web interface built with HTML, CSS, and Flask templates.
- Simple image upload form for blood cell images.
- Results page displays predicted blood cell type along with the image preview.
- Proper error handling for invalid or missing image files.

---

10. Testing

Testing Strategy:

- Manual testing was performed to ensure that each feature behaves as expected.
- Each functionality, including file upload, prediction, and error handling, was tested individually.
- Focus was on functional accuracy, UI clarity, and robustness under edge cases.

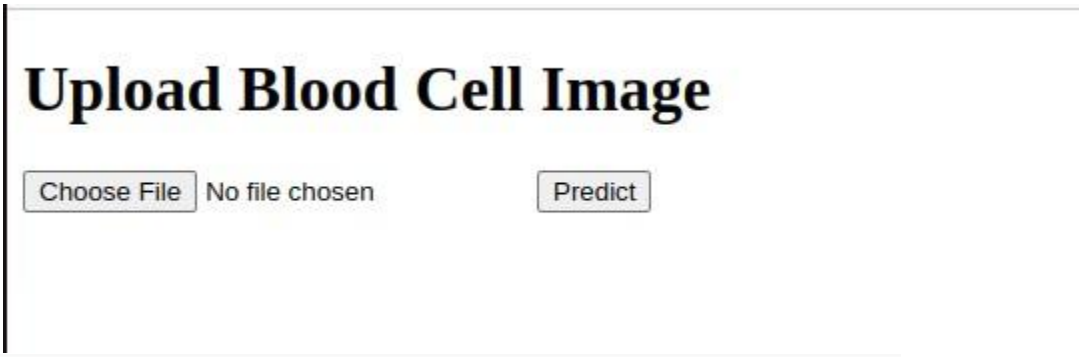
Test Cases Included:

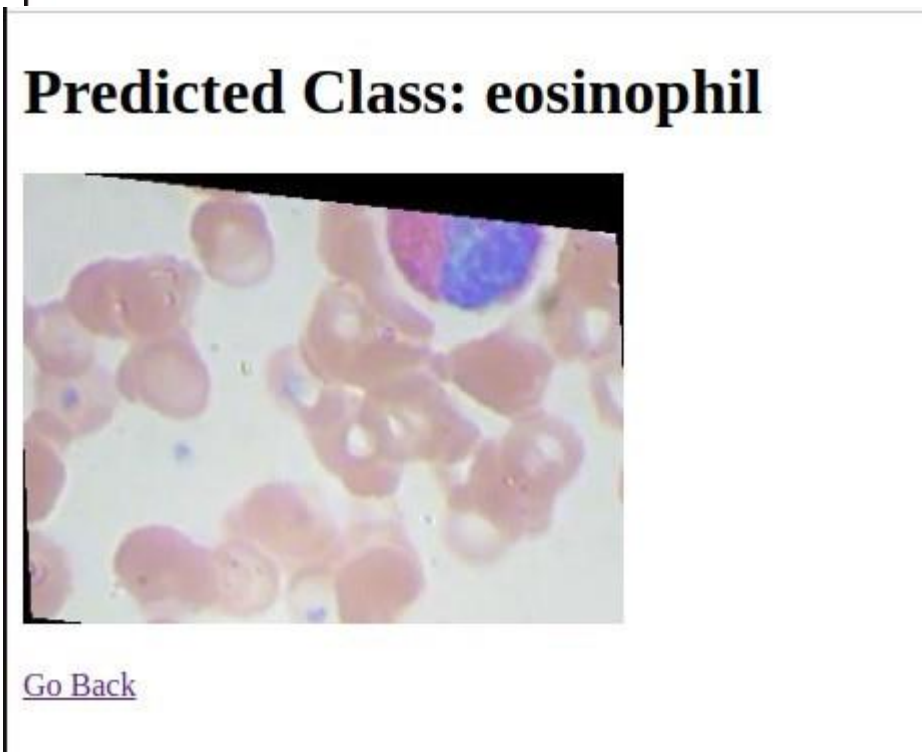
Test Case	Expected Result	Status
Upload valid blood cell image	Displays correct cell type prediction	☑ Pass
Upload invalid file (.txt)	Displays error message	☑ Pass
Upload very large image	Processes and predicts without crashing	☑ Pass
Submit with no image	Prompts user to select an image	☑ Pass
Model response time	Returns prediction within 2–3 seconds	☑ Pass

Tools Used:

- Web browser (Chrome/Firefox) for testing the user interface.
- Python `print` statements and `logging` for backend debugging.
- Manual uploads and form submissions to simulate user actions.

11. Screenshots or Demo





12. Known Issues

- No webcam or real-time camera support.
- Only supports the four trained blood cell classes: eosinophil, lymphocyte, monocyte, and neutrophil.
- No feedback mechanism or prediction logging.

- No user login system or prediction history tracking.

---

13. Future Enhancements

- ☑ Add mobile version or responsive UI for better accessibility.
- ☑ Integrate webcam for real-time blood cell detection.
- ☑ Expand dataset to improve prediction accuracy and generalization.
- ☑ Enable multilingual interface for broader usability in diverse regions.
- ☑ Add cloud database to store user feedback and analytics.
- ☑ Support classification of additional blood cell categories beyond the current four.