

## Hemaclassify: Web-based Blood Diseases Classification System Utilizing High-Resolution Neural Networks

Khushaalan Arjunan<sup>1</sup>, Afzan Adam<sup>2\*</sup>, Raja Zahratul Azma Raja Sabudin<sup>2</sup>, Mohammad Faizal Ahmad Fauzi<sup>3</sup>, Elaine Wan Ling Chan<sup>4</sup>

<sup>1</sup>Center for Artificial Intelligence Technology, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Malaysia

<sup>2</sup>Department of Pathology, UKM Medical Center, Universiti Kebangsaan Malaysia, Cheras, Kuala Lumpur, Malaysia

<sup>3</sup>Faculty of Engineering, Multimedia University, Cyberjaya, Malaysia

<sup>4</sup>Fusionex Ai Laboratoty, International Medical University, Kuala Lumpur, Malaysia

\* Corresponding author: afzan@ukm.edu.my

Received: 23 July 2024

Revised: 4 May 2025

Accepted: 24 June 2025

### ABSTRACT

*Blood classification plays a pivotal role in diagnosing various blood-related diseases, including Acute Lymphoblastic Leukemia (ALL), Acute Promyelocytic Leukemia (APML), Chronic Myeloid Leukemia (CML), Iron Deficiency Anemia (IDA), Thalassemia Major, Thalassemia Minor, and Normal Blood Cells. Manual blood analysis by hematologists is time-consuming and often lacks the speed needed for critical diagnoses. To improve efficiency, deep learning and image processing techniques are applied. This project aims to address these challenges by developing a web-based expert system for the automatic classification of blood samples using High-Resolution Neural Networks (HRNet). The system maintains high-resolution representations throughout the classification process. An innovative approach connects high-to-low resolution convolution streams in parallel, facilitating the preservation of high-resolution information. Cross-resolution information exchange improves semantic richness and spatial precision. The system is capable of classifying blood samples into seven distinct categories as mentioned earlier. The proposed web application streamlines the diagnostic process, offering a faster and more accurate classification of blood samples. Automation aids hematologists and helps prevent CML from progressing to acute stages. The total image dataset used in this project is 690 which was divided into a ratio of 8:1:1, representing the training, validation, and testing datasets. The total number of images used for training, validation, and testing are 549, 66, and 75, respectively. The system achieved an accuracy rate of 99.89%, demonstrating its effectiveness in blood classification. HemaClassify uses REST architecture, with a frontend in NextJS, TailwindCSS, ShadCN UI, and a Flask API backend using HRNet. MySQL is used for database management. User Acceptance Testing by technologists and hematologists at UKM Medical Center confirmed it meets end-user needs. Most users rated usability and functionality 4/5 or 5/5, showing high satisfaction. This project represents a significant advancement in the field of blood classification, contributing to more timely and precise diagnoses, which are essential for effective disease management and treatment.*

**Keywords:** Blood Classification, HRNet, Deep Learning, Web-based System, Medical Diagnosis

## 1 INTRODUCTION

The identification and classification of blood cells are crucial for accurate diagnosis and effective intervention in various blood-related diseases. Traditional methods used by hematologists for blood classification have limitations such as subjectivity, time inefficiency, and potential diagnostic errors. Accurate blood classification is vital, impacting treatment planning and patient outcomes. In acute conditions like Acute Lymphoblastic Leukemia (ALL), Acute Promyelocytic Leukemia (APML), and Chronic Myeloid Leukemia (CML) [1], precise diagnosis is essential for initiating timely therapeutic strategies. Similarly, early identification in chronic diseases like thalassemia aids in managing these lifelong disorders. Developing automated systems for blood classification holds the potential to revolutionize diagnostic processes, enhancing efficiency and accuracy [2]. This project proposes the use of High-Resolution Neural Networks (HRNet) [3] for automatic blood sample classification, leveraging deep learning and image processing techniques to extract intricate features distinguishing seven blood categories. The system will be implemented using PyTorch with CUDA support for efficient model training and will be trained on blood biopsy samples from UKM Hospital. The resulting web application aims to serve both hematologists and non-experts, featuring a user-friendly interface for seamless interaction. The objective is to develop a web-based application using HRNet classification models to automatically classify blood samples into seven categories, targeting an accuracy rate of 95% or higher [4]. The final product will be a REST architecture-based [5] web application, trained and tested with images from UKM Hospital, designed to support and expedite the diagnostic process for hematologists while adhering to ethical data handling practices (JEP-2022-317). The project will adopt an iterative development model to accommodate flexibility and minimize risks associated with potential uncertainties. Challenges such as data confidentiality concerns, limited access to expert hematologists for validation, and the inherent complexity of the domain will be carefully addressed through ethical data handling practices and expert collaboration. The project schedule is outlined for two semesters, ensuring systematic and goal-oriented development in line with SMART criteria to facilitate the creation of a web-based system accessible across borders.

## 2 MATERIAL AND METHODS

The development model chosen for this project is an Iterative and Incremental development model [6]. This model allows for iterative refinement, accommodates flexibility, and reduces risks associated with the uncertainty of potential project requirements. The decision to choose incremental development is based on its adaptability to changing project dynamics and the iterative nature of developing sophisticated systems.

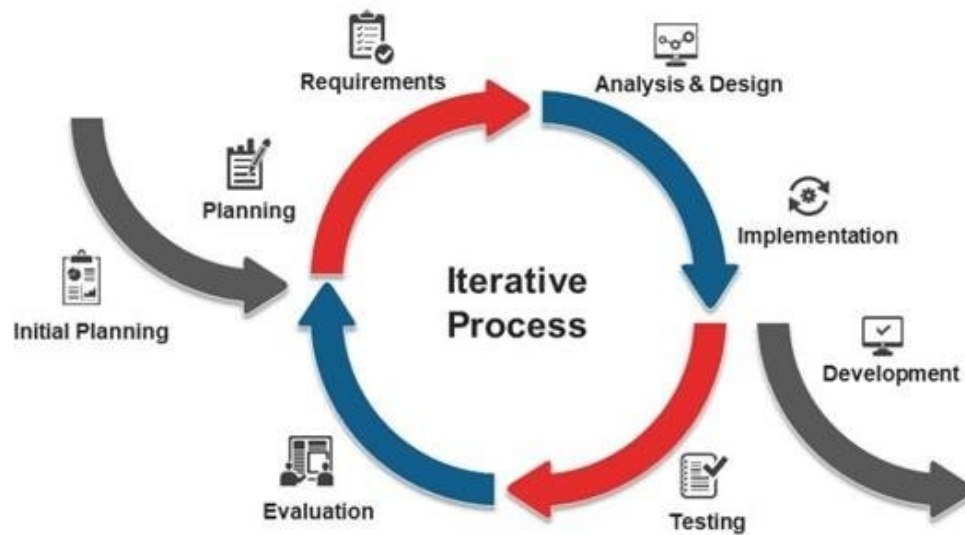


Figure 1 : Iterative and Incremental Development Model

## 2.1 Planning Phase

The planning phase involves defining the project scope, objectives, and requirements. During this phase, the problem and solution for that problem are identified, and the project's title is determined. Once the project scope is defined, a detailed project plan is developed, outlining the project timeline for the development, testing, and deployment phases. The planning phase sets the foundation for subsequent development activities and ensures alignment with project goals.

## 2.2 Analysis Phase

The analysis phase focuses on gathering and analyzing user requirements, system specifications, and constraints. During this phase, literature reviews were conducted to identify existing solutions in the market and comparisons were made to determine the unique selling proposition of the proposed system. User Requirement and System Requirement Specifications were developed to define the system's functional and non-functional requirements. The analysis phase lays the groundwork for designing the system architecture and developing the solution.

## 2.3 Design Phase

The design phase involves creating detailed system specifications, architectural diagrams, and user interface mockups. During this phase, the Use Case Diagram, Use Case Specification, Class Diagram, Sequence Diagram, and Architecture Diagram were developed. The design phase ensures that the system architecture aligns with project objectives and user needs.

## 2.4 Implementation Phase

The implementation phase focuses on developing the system components, and testing the system's functionality. During this phase, the dataset was preprocessed and split into training, validation,

and testing sets. The HRNet model was trained using the training set and validated using the validation set. The Server Side and Client Side were developed using Flask and Next.js, respectively. The database was implemented using MySQL, and the system was containerized using Docker. The system was deployed on the Medical Computing Lab server. The implementation phase involves iterative development cycles to refine the system based on feedback and testing results.

## **2.5 Testing Phase**

The testing phase involves validating the system's functionality, performance, and usability [7]. During this phase, both functional and non-functional tests were conducted. Functional tests verify that the system meets the specified requirements, while non-functional tests assess the system's performance, and reliability. The Functional Test conducted includes Unit Testing, Integration Testing, API Testing, and User Acceptance Testing. Load Testing were conducted as part of Non-Functional Testing. The User Acceptance Testing (UAT) were conducted from 1st July 2024 to 7th July 2024 where the end users were invited to test the system at <http://hemaClassify.ukm.my>. The end users for the test include Medical Laboratory Technologists and Hematologists from Hospital UKM, under the leadership of Dr. Raja Zahratul Azma Raja Sabudin, the head of the department. Additionally, Faculty of Information Science & Technology (FTSM) students tested the system in the admin role. Once the test was completed, the feedback was collected and analyzed to determine the system's usability and functionality whether all the requirements were met. The survey was conducted using Google Forms and the results were analyzed using Google Sheets. After the User Acceptance Testing, the system was corrected based on the feedback received. The testing phase identifies and resolves defects, refines system features, and prepares the system for deployment.

## **3 RESULTS AND DISCUSSION**

### **3.1 System Architecture**

HemaClassify uses Representational State Transfer (REST) architecture, which is a stateless client-server architecture that allows for scalability and flexibility. The Server side is responsible for handling requests and responses, while the Client side is responsible for fetching and displaying data. The NGINX server in the middle acts as a load balancer, distributing requests to the appropriate server [8]. Below is the architecture diagram for the HemaClassify system:

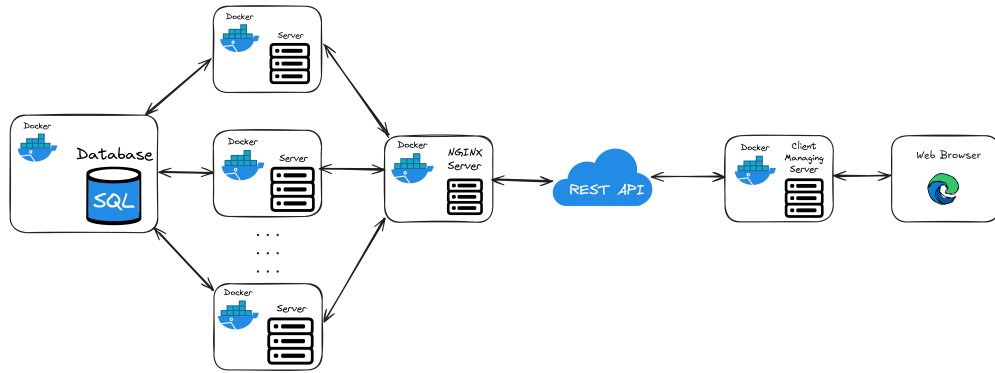


Figure 2 : HemaClassify Architecture Diagram

### 3.2 Model Training

The HRNet model was trained for 500 epochs, taking a total of 11.31 hours using a single NVIDIA GeForce RTX 3060 GPU. Firstly, HemaClassify was trained for 300 epochs, taking 8.26 hours. The train\_top1 and train\_loss values were 99.6561 and 0.0282, respectively. Below is the visualization for Train Top and Train Loss from epochs 0-299:

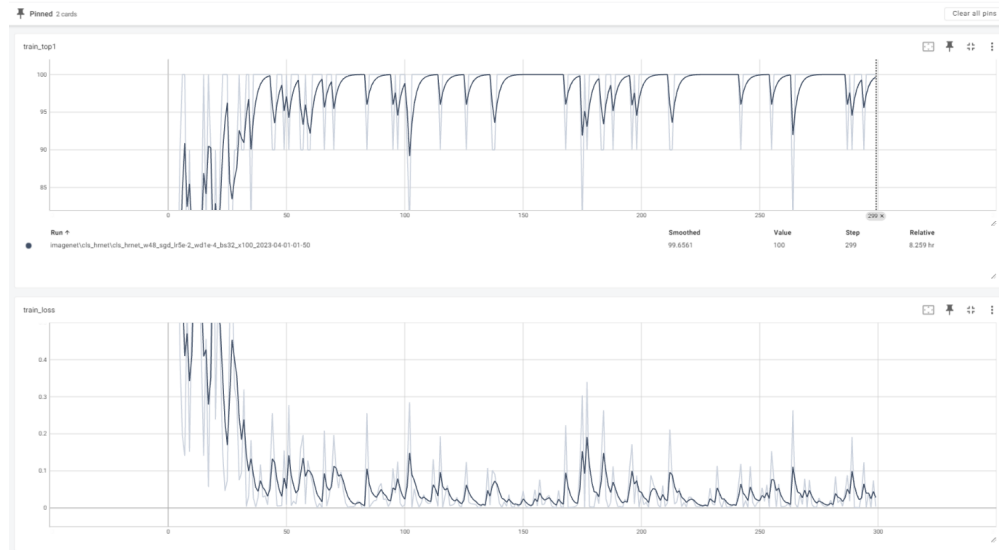


Figure 3 : Iterative and Incremental Model

Then, HemaClassify was trained for an additional 200 epochs, taking 3.05 hours. The train\_top1 and train\_loss values were 99.8867 and 0.0189, respectively. Below is the visualization for Train Loss from epochs 300-499:



Figure 4 : Iterative and Incremental Model

This indicates that the HRNet model has been well-trained and can accurately predict the class of blood samples. The high train\_top1 value indicates that the HRNet model has high accuracy. The low train\_loss value indicates that the HRNet model has low error. The HRNet model architecture maintains high-resolution representations throughout the network, enabling precise feature extraction. The key components of HRNet can be described mathematically as follows:

- **Parallel Multi-Resolution Convolutions [3] [9]** : HRNet connects multiple convolution streams in parallel, each operating at a different resolution. Let  $x$  denote the input image, and  $f_k(x)$  represent the feature map at the  $k$ -th resolution level, where  $k \in \{1, 2, \dots, K\}$ . The feature maps at each resolution are computed as:

$$f_k(x) = \text{Conv}_k(x), \quad (1)$$

where  $\text{Conv}_k$  denotes the convolution operation at the  $k$ -th resolution level.

### 3.3 Server Side Development

The HemaClassify server side was developed using Flask [10], a micro web framework written in Python. The server side has a total of 20 endpoints. These endpoints can be categorized into 4 categories: Authentication, User, Patient, and Image.

- **Authentication**

- `/getjwtaccesstoken` - GET - Obtain JWT token for authentication

- **User**

- `/user` - GET - List Users
- `/user` - POST - Add User

- /user - PUT - Update User
- /user - DELETE - Delete User
- /user?uname= - GET - User Information

- **Patient**

- /patient - GET - List Patients
- /patient - POST - Add Patient
- /patient - PUT - Update Patient
- /patient - DELETE - Delete Patient
- /patient?patient\_id= - GET - Patient Information
- /patient?patient\_id\_status= - GET - Information on Total Images and Unverified Images

- **Image**

- /image?patient\_id= - GET - List Patient Images
- /image?image\_id= - GET - Image Information
- /image - POST - Add Image
- /image - DELETE - Delete Image
- /image?unverified-image= - GET - List of Unverified Images
- /image?unverified-image-count= - GET - Number of Unverified Images
- /image - PUT - Update Final Image Result
- /image?uncached-image= - GET - Uncached Images

HemaClassify uses JSON Web Token (JWT) for authentication during login. The algorithm used for encoding the information is HS256 [11] . After authentication is successful, it stores the access token in local storage. For each subsequent API request, the HemaClassify Client needs to send the access token in the headers to obtain authorization. Below is the example of a request to the /user endpoint with the access token in the headers:

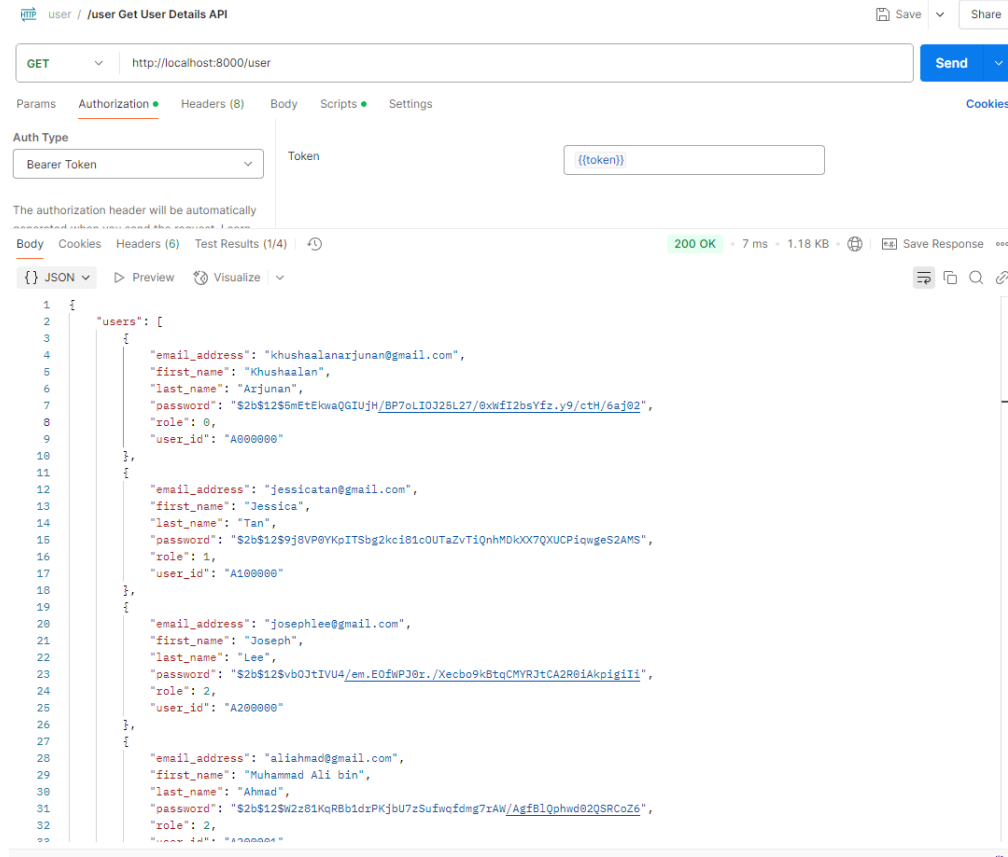


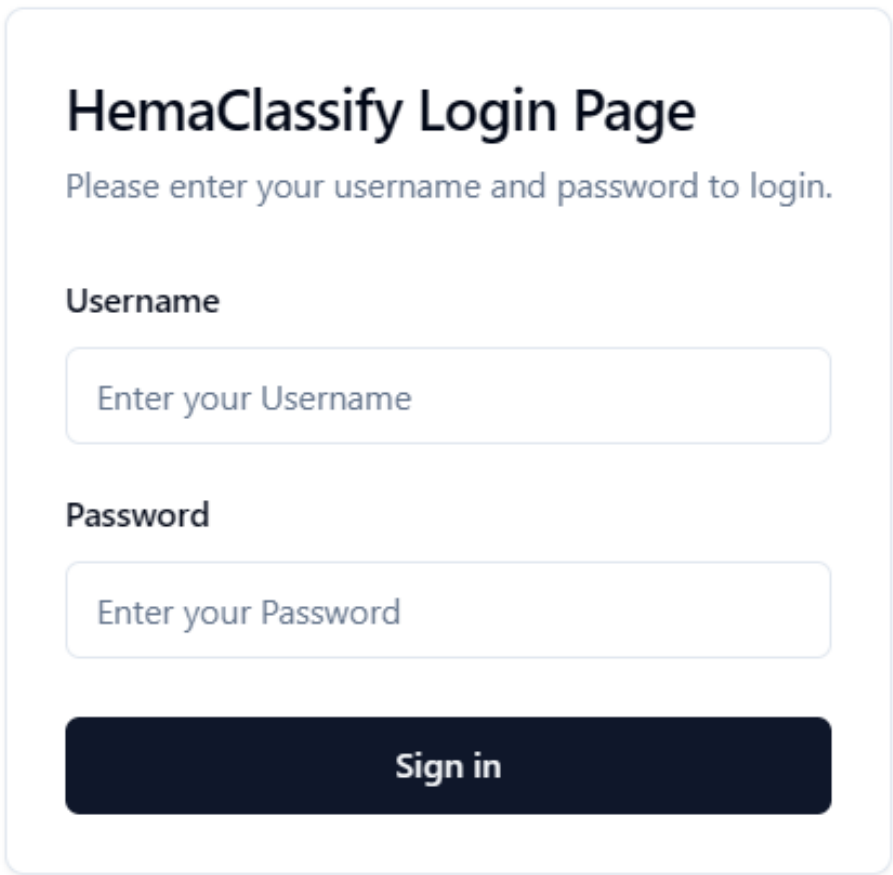
Figure 5 : REST API Request

### 3.4 Client Side Development

The HemaClassify client side uses Next.js framework version 14.2.4 with TypeScript, Tailwind CSS [12], and shadcn UI [13]. HemaClassify employs both Server Side and Client Side Rendering. HemaClassify uses File Based Routing where each folder under `/src/app/` represents a separate route. For instance, HemaClassify has folders like `/dashboard`, `/users`, `/patients`, and `/unverifiedimages`, each containing a `page.tsx` file that renders TypeScript components. The `layout.tsx` file provides the Root Layout for the HemaClassify application. React elements are extensively used and refactored into small components, such as `Navbar` and `Footer`, which are reused across routes. React Hooks like `useState`, `useEffect`, `useRef`, are used to manage state and side effects in functional components, making component programming more intuitive and efficient. HemaClassify utilizes Tailwind CSS and Shadcn UI for its user interface design. Tailwind CSS is recognized for its utility-first approach, which simplifies development by providing reusable classes that can be combined to create complex designs easily. This approach replaces traditional CSS, allowing direct customization of elements in HTML markup, resulting in more consistent styles and cleaner code. Using utility classes also accelerates development by eliminating the need for extensive and repetitive CSS writing. ShadCN offers pre-designed UI components ready for production. These components are meticulously crafted and tested to ensure they can be seamlessly integrated with



minimal adjustments. This capability allows HemaClassify to implement high-quality, consistent UI components swiftly, ensuring visually appealing and functional interfaces. Below are some screenshots of the HemaClassify web interface:



**HemaClassify Login Page**

Please enter your username and password to login.

**Username**

**Password**

**Sign in**

---

© 2024/2025 HemaClassify. All rights reserved.

---

Figure 6 : HemaClassify Login

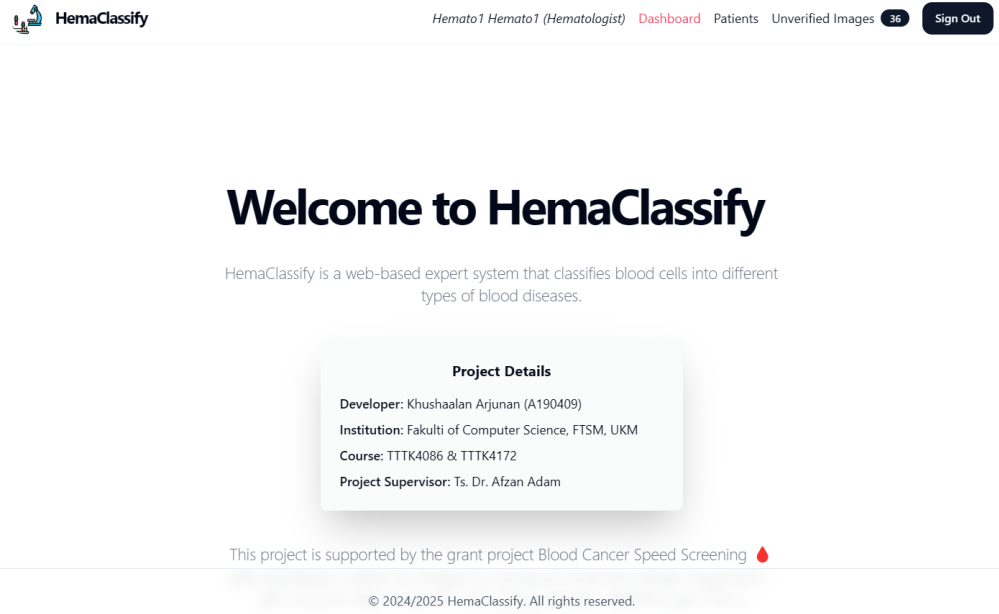


Figure 7 : HemaClassify Dashboard

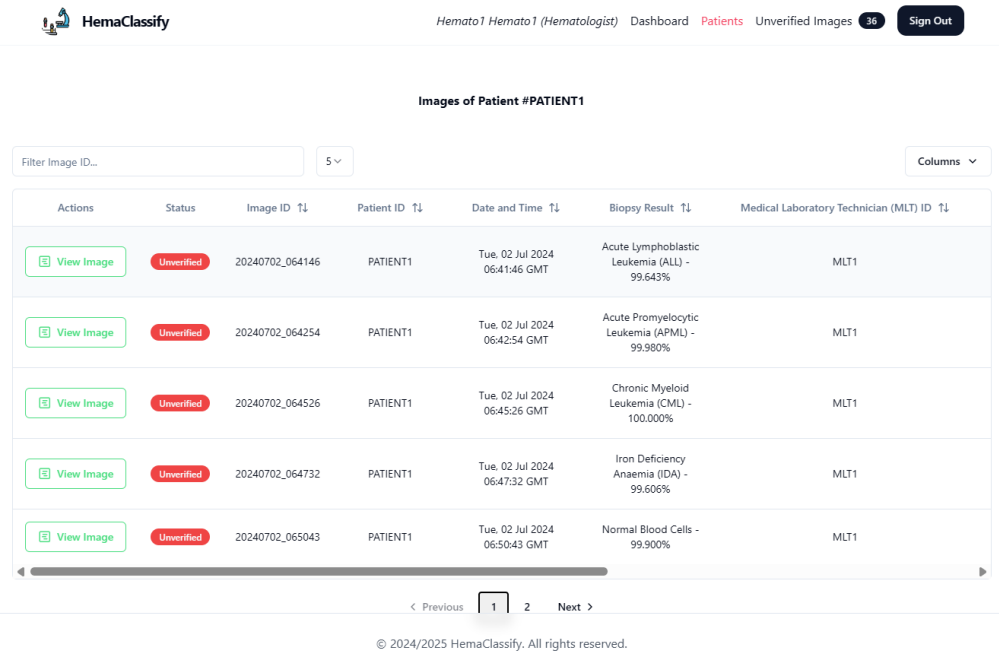


Figure 8 : HemaClassify Images

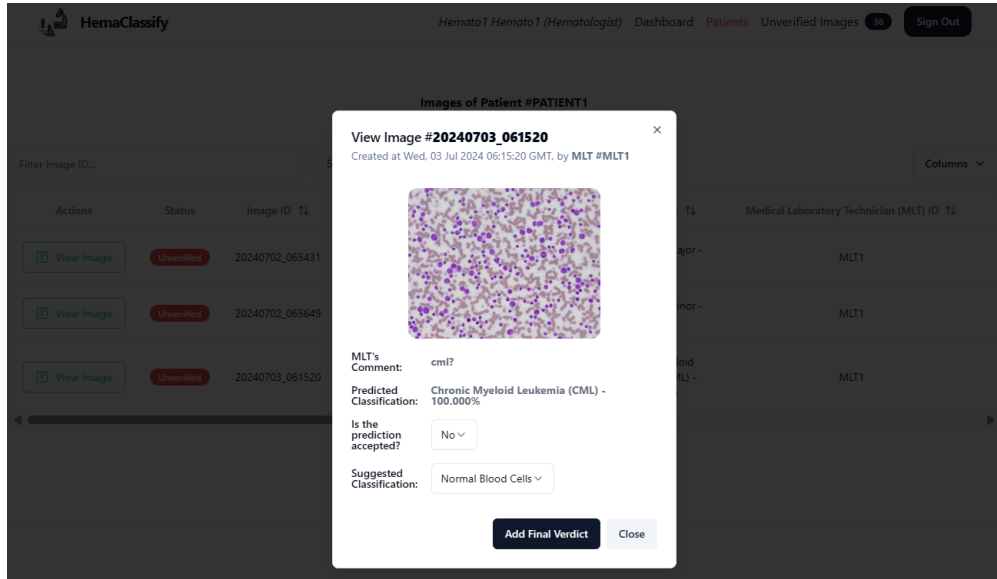


Figure 9 : HemaClassify Image

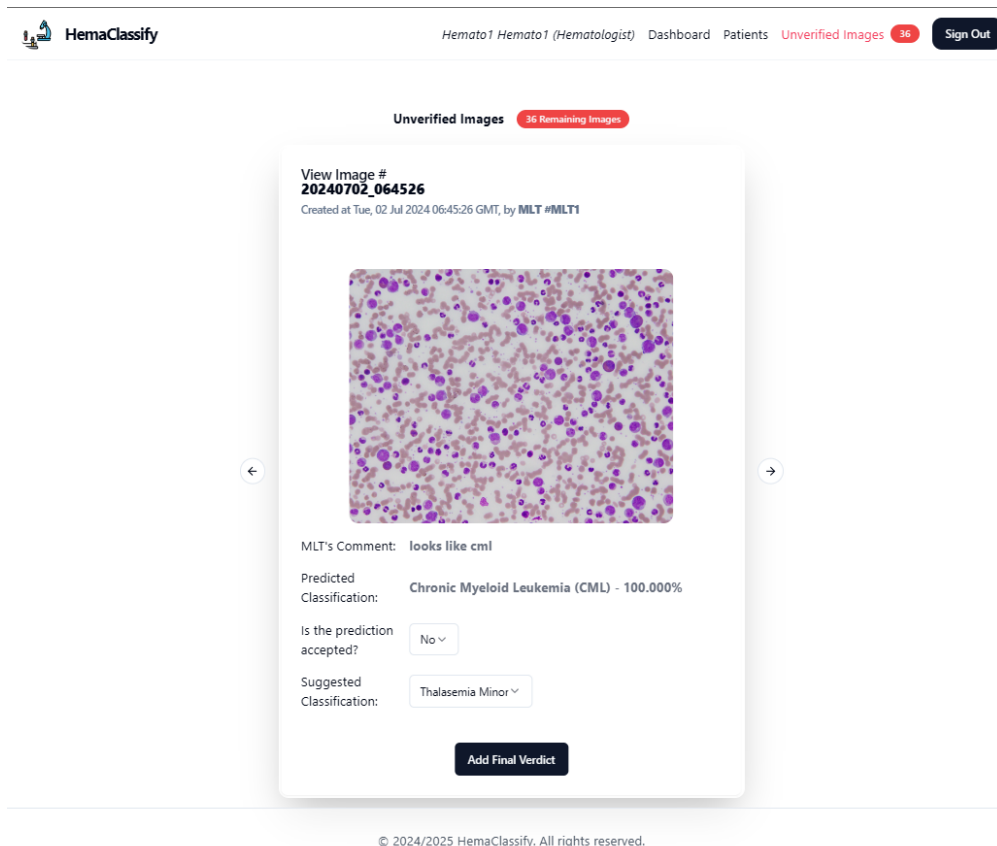


Figure 10 : HemaClassify Unverified Images

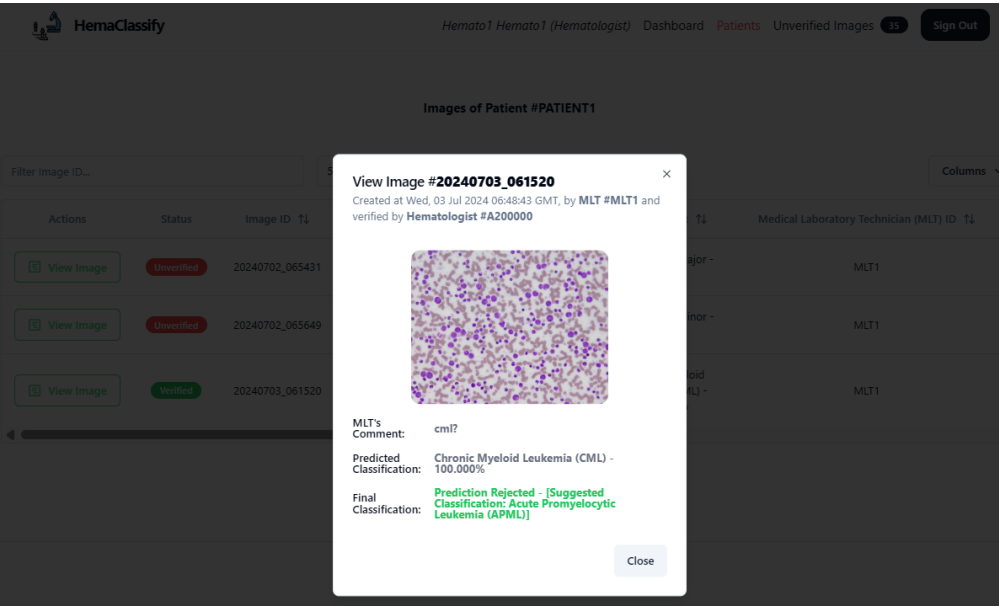


Figure 11 : HemaClassify Verified Image

### 3.5 Database Management

HemaClassify uses MySQL for database management. The database consists of 3 table which are `tbl_users_a190409`, `tbl_patients_a190409`, and `tbl_images_a190409`. Below is the class diagram for the database schema:

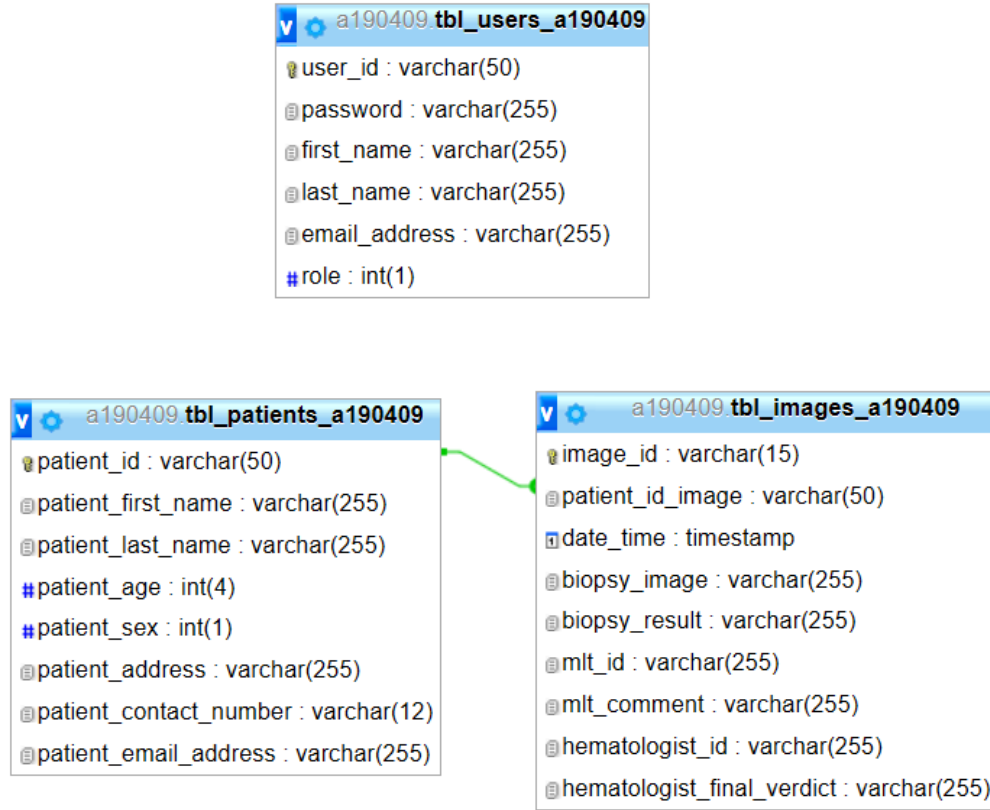


Figure 12 : Database Class Diagram

### 3.6 Containerization and Deployment

HemaClassify also utilizes Docker technology for containerization. Containerization is crucial in web application development. Docker allows developers to build, test, and deploy applications easily. To create Docker containers, a Docker image needs to be built first, which defines the container's configuration using a Dockerfile. Once the Docker image is built, Docker run commands are used to create Docker containers. For managing multiple containers, Docker Compose can be used. HemaClassify employs four Docker containers: hemaclassify-server, hemaclassify-client, mysql, and nginx. The hemaclassify-server and hemaclassify-client containers are for the server and client parts respectively. The mysql container is for the MySQL database, and the nginx container is for the web server. Configuration for all containers is defined in the docker compose.yml file. After containerization with Docker, HemaClassify was deployed on the Medical Computing Lab server. It has obtained a Public IP and domain name, accessible via <http://hemaclassify.ukm.my>. However, HemaClassify currently lacks an SSL Security Certificate, so it can only be accessed via HTTP using UKM Wi-Fi.

### 3.7 Functional and Non-Functional Test Results

Unit Test was written in Python using the unittest library. The test cases were written to test the functionality of the HRNet model and hashing function for password encryption. Integration Test

was written using Pytest to test the integration of database operations like CRUD (Create, Read, Update, Delete) and the Flask API.

```
khushaalan@khushaalan MINGW64 ~/Desktop/semester6/TTTK4086/hemaclassify/server (main)
• $ PYTHONWARNINGS="ignore" python -m unittest discover -v
test_hashing_admin_correct_password (test_unittest.TestHashingFunction) ... ok
test_hashing_admin_incorrect_password (test_unittest.TestHashingFunction) ... ok
test_hashing_hematologist_correct_password (test_unittest.TestHashingFunction) ... ok
test_hashing_hematologist_incorrect_password (test_unittest.TestHashingFunction) ... ok
test_hashing_mlt_correct_password (test_unittest.TestHashingFunction) ... ok
test_hashing_mlt_incorrect_password (test_unittest.TestHashingFunction) ... ok
test_predict_ALL (test_unittest.TestPredictionFunction) ... ok
test_predict_APLM (test_unittest.TestPredictionFunction) ... ok
test_predict_CML (test_unittest.TestPredictionFunction) ... ok
test_predict_IDA (test_unittest.TestPredictionFunction) ... ok
test_predict_NORMAL (test_unittest.TestPredictionFunction) ... ok
test_predict_TMAJOR (test_unittest.TestPredictionFunction) ... ok
test_predict_TMINOR (test_unittest.TestPredictionFunction) ... ok

-----
Ran 13 tests in 17.582s

OK
```

Figure 13 : Unit Test Result

```
khushaalan@khushaalan MINGW64 ~/Desktop/semester6/TTTK4086/hemaclassify/server (main)
• $ pytest test_auth.py test_user.py test_patient.py test_image.py
===== test session starts =====
platform win32 -- Python 3.10.11, pytest-8.3.5, pluggy-1.5.0 -- C:\Users\khushaalan\AppData\Local\Programs\Python\Python310\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\khushaalan\Desktop\semester6\TTTK4086\hemaclassify\server
configfile: pytest.ini
collected 19 items

test_auth.py::test_auth PASSED [ 5%]
test_user.py::test_get_user PASSED [ 10%]
test_user.py::test_get_users PASSED [ 15%]
test_user.py::test_add_user PASSED [ 21%]
test_user.py::test_edit_user PASSED [ 26%]
test_user.py::test_delete_user PASSED [ 31%]
test_patient.py::test_get_patient PASSED [ 36%]
test_patient.py::test_get_patients PASSED [ 42%]
test_patient.py::test_add_patient PASSED [ 47%]
test_patient.py::test_edit_patient PASSED [ 52%]
test_patient.py::test_delete_patient PASSED [ 57%]
test_image.py::test_add_image PASSED [ 63%]
test_image.py::test_get_image PASSED [ 68%]
test_image.py::test_get_images PASSED [ 73%]
test_image.py::test_update_final_verdict PASSED [ 78%]
test_image.py::test_unverified_images PASSED [ 84%]
test_image.py::test_unverified_images_count PASSED [ 89%]
test_image.py::test_get_uncached_image PASSED [ 94%]
test_image.py::test_delete_image PASSED [100%]

===== 19 passed in 4.17s =====
```

Figure 14 : Integration Test Result

API Test was written in Node.js using Jest to test the API endpoints. The test cases were written to test the functionality of the API endpoints, including authentication, user, patient, and image operations.

```
khushaalan@khushaalan MINGW64 ~/Desktop/semester6/TTTK4086/hemaclassify/client2 (main)
$ npm test

> client2@0.1.0 test
> jest

PASS tests/api.test.ts (5.597 s)
  HemaClassify Authentication
    ✓ GET /getjwtaccesstoken returns a JWT access token (258 ms)
  HemaClassify User
    ✓ GET /user returns user list (23 ms)
    ✓ POST /user adds a new user (231 ms)
    ✓ PUT /user updates a user (10 ms)
    ✓ GET /user?uname= returns user info (9 ms)
    ✓ DELETE /user deletes a user (13 ms)
  HemaClassify Patient
    ✓ GET /patient returns patient list (13 ms)
    ✓ POST /patient adds a new patient (10 ms)
    ✓ PUT /patient updates a patient (9 ms)
    ✓ GET /patient?patient_id= returns patient info (9 ms)
    ✓ GET /patient?patient_id_status= returns patient total images and total unverified images (8 ms)
    ✓ DELETE /patient deletes a patient (13 ms)
  HemaClassify Image
    ✓ POST /image adds a new image (2981 ms)
    ✓ GET /image?patient_id= returns image list (11 ms)
    ✓ GET /image?image_id= returns image info (11 ms)
    ✓ GET /image?unverified-image=1 returns unverified image list (16 ms)
    ✓ GET /image?unverified-image-count=1 returns unverified image count (12 ms)
    ✓ GET /image?uncached-image=1 returns uncached image list (6 ms)
    ✓ PUT /image add final verdict (11 ms)
    ✓ DELETE /image deletes an image (21 ms)

Test Suites: 1 passed, 1 total
Tests: 20 passed, 20 total
Snapshots: 0 total
Time: 5.706 s, estimated 7 s
Ran all test suites.
```

Figure 15 : API Test Result

Load Test was conducted using Apache JMeter to test the system's performance under various loads. Requests were sent to 10 API endpoints simultaneously with 300 threads and a 1s ramp-up period. The system did not fail at all. The error percentage is 0%. However, the response time increases as user load increases. This indicates that the system still requires improvements in scaling and reducing response times to enhance system performance.

	Label	# Samples	Average	Min	Max	Std. Dev	Error %	Throughput /s	Received KB/sec	Sent KB/sec	Avg. Bytes
GET /user		300	43645	8448	78736	20367.71	0.00%	3.4/sec	3.20	1.82	960.8
GET /getjwtaccesstoken		300	39633	300	78936	22768.76	0.00%	3.8/sec	2.32	0.58	831.9
GET /user?uname=		300	8690	8410	55908	3104.42	0.00%	4.3/sec	1.61	2.35	386.0
GET /image/unverified-image-count=		300	16617	11221	16679	355.16	0.00%	7.7/sec	1.40	4.31	187.9
GET /image/uncached-image=&image_id=		300	12039	7359	16676	2708.42	0.00%	8.5/sec	36239.59	4.93	4345998.0
GET /image/unverified-image=		300	12601	8493	16650	2350.81	0.00%	9.7/sec	7.90	5.38	835.0
GET /image?patient_id=		300	8548	8487	8720	47.17	0.00%	13.1/sec	10.81	7.34	846.0
GET /patient?patient_id=		300	8657	8473	8734	35.90	0.00%	13.1/sec	5.48	7.38	429.0
GET /image?image_id=		300	8506	8470	8594	27.95	0.00%	13.1/sec	6.79	7.35	530.0
GET /patient		300	8493	8429	8601	54.07	0.00%	13.2/sec	5.72	7.10	444.0
TOTAL		3000	16743	300	78936	16050.49	0.00%	17.5/sec	7429.95	9.01	435124.0

Figure 16 : JMeter Report

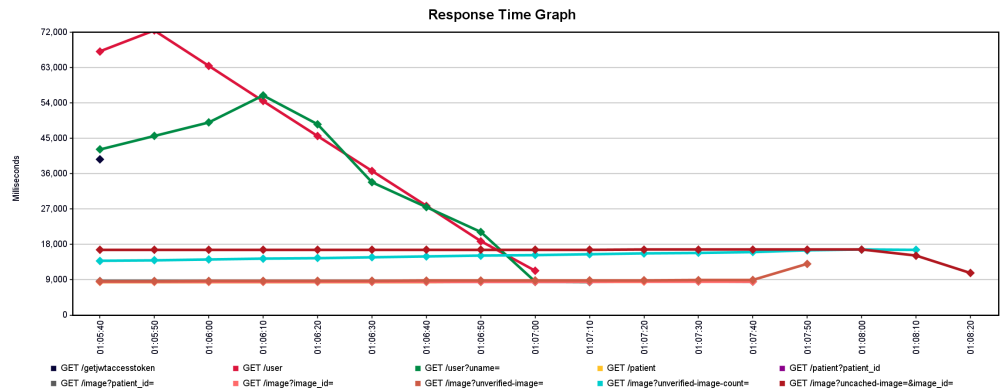


Figure 17 : JMeter Response Time

User Acceptance Testing’s survey results indicate a high level of user satisfaction with the HemaClassify system. Most users rated the system 4/5 or 5/5 for its usability and functionality. The system met all the requirements and expectations of end-users, demonstrating its effectiveness in blood classification.

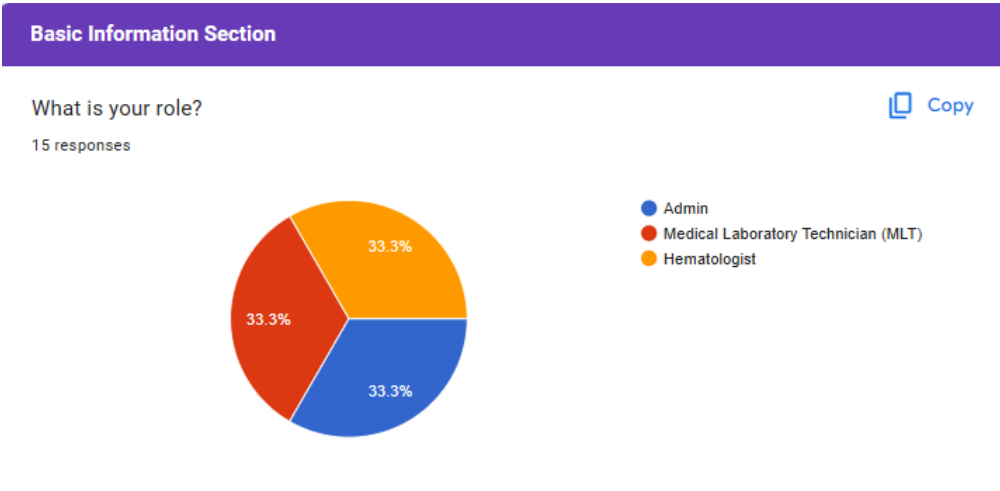


Figure 18 : User Acceptance Testing Result

A	B	C	D	E	F	G
What is your role?	Were you able to view all users and utilize the filtering, searching, and sorting functionalities effectively?	Were you able to add, edit, update, and delete users without any issues?	Did you find the system is intuitive, so that you can guess how to use it without extensive instructions?	How user-friendly did you find the interface for managing users?	How would you rate the overall look and feel of the system?	Additional Comments or Suggestions
Admin	5	Yes	5	5	5	
Admin	4	Yes	5	5	5	
Admin	5	Yes	5	5	5	
Admin	5	Yes	5	5	5	
Admin	5	Yes	5	4	5	

Figure 19 : User Acceptance Testing Result



A	M	N	O	P	Q
What is your role?	Were you able to view image detail and add final verdicts without any issues?	Did you find the system intuitive to use without extensive instructions?	How user-friendly did you find the interface for managing patients, images, and unverified images?	How would you rate the overall look and feel of the system? (e.g., smart, clean, modern)	Additional Comments or Suggestions
Hematologist	Yes	5	5	5	Excellent
Hematologist	Yes	4	3	4	The system interface is quite lag behind. It takes time to go through from one image to another image. Therefore, the developer should improve the system interface to increase the system efficacy & save the time. Otherwise, this is a good initiative for the slide review at the high through put centre prior to further morphology assessment by the haematologist.
Hematologist	No	5	4	3	Simple intuitive system to use
Hematologist	Yes	5	5	5	-
Hematologist	Yes	5	5	5	-

Add 1000 more rows at the bottom

Figure 20 : User Acceptance Testing Result

A	G	H	I	J	K	Q
What is your role?	Were you able to view, filter, search, and sort the patient list and image list effectively?	Were you able to add, edit, update, and delete patients and images without any issues?	Did you find the system is intuitive, so that you can guess how to use it without extensive instructions?	How user-friendly did you find the interface for managing patients and images?	How would you rate the overall look and feel of the system? (e.g., smart, clean, modern)	Additional Comments or Suggestions
Medical Laboratory Technician (MLT)	5	Yes	5	5	5	
Medical Laboratory Technician (MLT)	5	Yes	5	5	4	
Medical Laboratory Technician (MLT)	5	Yes	5	5	5	
Medical Laboratory Technician (MLT)	5	Yes	5	5	5	Allow MLT to upload multiple images simultaneously
Medical Laboratory Technician (MLT)	5	Yes	5	5	5	Great job! Impressed by the library choice for the GUI. Sleek!

Add 1000 more rows at the bottom

Figure 21 : User Acceptance Testing Result

### 3.8 Recommendations for Improvement

To enhance the performance and capabilities of HemaClassify, several improvement recommendations are proposed. The first recommendation is that HemaClassify can implement multiple layers of caching using Redis and server-side pagination to improve system performance. The second recommendation is to use a dedicated database server such as LRGS FTSM or a cloud-based database server like AWS RDS to facilitate data migration and backup processes. The third recommendation is for HemaClassify to use deep learning technologies like OpenSlide [14] to automatically process Whole Slide Images (WSI) [15] into Regions of Interest (ROI) [16]. The fourth recommendation is to add a function that allows users to upload their own datasets and train new models or use existing models for transfer learning. The final recommendation is for HemaClassify to register an SSL certificate to use the HTTPS protocol, enhancing system security and accessibility.

## 4 CONCLUSION

In conclusion, HemaClassify has been successfully developed and met its main objective of identifying blood diseases with high accuracy. Despite some constraints, the system has shown great potential in aiding the diagnostic process. With the proposed improvements, HemaClassify has the potential to become a more comprehensive and effective tool in the field of hematology. It is hoped that this system can continue to be developed and fully utilized by medical experts and clinical practitioners to improve the accuracy and efficiency of blood disease diagnosis.

### 4.1 Strengths

HemaClassify was successfully built, tested, and deployed within the designated timeframe without any issues. The system meets all the objectives and user requirements set forth. HemaClassify can identify 7 categories of blood diseases with high accuracy. The system is user-friendly and easy to use. HemaClassify can operate on both mobile devices and desktop computers. It can be accessed by users from anywhere and at any time at <http://hemaclassify.ukm.my>.

### 4.2 Weaknesses

There are several constraints that need to be addressed in the HemaClassify system. Firstly, the interactivity of HemaClassify is not yet smooth and needs improvement. For example, when there are many unverified images, the image carousel does not move smoothly. This is due to the large number of high resolution images being fetched from the server at once. The second constraint is that HemaClassify's database is still running in a Docker container. This setup is suitable for mock projects but very challenging when it comes to large-scale migration or data backup. The third constraint is that HemaClassify cannot yet process large Whole Slide Images (WSI) into smaller Regions of Interest (ROI). The crop and zoom functions are also not yet available. The fourth constraint is that HemaClassify is limited to only 7 categories of blood diseases and cannot identify other blood diseases. The final constraint is that although HemaClassify has its own domain name, it still uses the HTTP protocol, which may be blocked by some Internet Service Providers (ISPs).

## ACKNOWLEDGEMENT

This project was supported by the Blood Cancer Speed Screening grant (KKP/2020/MMU-UKM/7/2) funded by the Kementerian Pendidikan Tinggi (KPT). All image data were taken from UKM Medical Center with research ethics code JEP-2022-317 granted by UKM Research Ethics Committee.

## REFERENCES

- [1] C. L. Sawyers, "Chronic myeloid leukemia," *New England Journal of Medicine*, vol. 340, no. 17, pp. 1330–1340, 1999, PMID: 10219069. [Online]. Available: <https://doi.org/10.1056/NEJM199904293401706>
- [2] A. Zuraw, *Digital Pathology 101: All You Need to Know to Start and Continue Your Digital Pathology Journey*. Independently published, October 5 2023, hardcover. [Online]. Available: <https://www.amazon.com/dp/B0CKJCTBNN>

- [3] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, “Deep high-resolution representation learning for visual recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3349–3364, 2021.
- [4] J.-N. Eckardt, T. Schmittmann, S. Riechert, M. Kramer, A. S. Sulaiman, K. Sockel, F. Kroschinsky, J. Schetelig, L. Wagenführ, U. Schuler, U. Platzbecker, C. Thiede, F. Stölzel, C. Röllig, M. Bornhäuser, K. Wendt, and J. M. Middeke, “Deep learning identifies acute promyelocytic leukemia in bone marrow smears,” *BMC Cancer*, vol. 22, no. 1, p. 201, 2 2022. [Online]. Available: <https://doi.org/10.1186/s12885-022-09307-8>
- [5] D. V. Kornienko, S. V. Mishina, S. V. Shcherbatykh, and M. O. Melnikov, “Principles of securing restful api web services developed with python frameworks,” *Journal of Physics: Conference Series*, vol. 2094, no. 3, p. 032016, Nov. 2021. [Online]. Available: <http://dx.doi.org/10.1088/1742-6596/2094/3/032016>
- [6] C. Larman and V. R. Basili, “Iterative and incremental developments. a brief history,” *Computer*, vol. 36, pp. 47–56, 2003. [Online]. Available: <https://api.semanticscholar.org/CorpusID:9240477>
- [7] P. C. Jorgensen, *Software Testing: a Craftsman’s Approach*, 4th ed. Auerbach Publications, 2013.
- [8] A. Chyrvon, K. Lisovskyi, and N. Kyryndas, “The main methods of load balancing on the nginx web server,” in *Scientific Practice: Modern and Classical Research Methods*. European Scientific Platform, May 2023. [Online]. Available: <http://dx.doi.org/10.36074/logos-26.05.2023.040>
- [9] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, “Medical image classification with convolutional neural network,” in *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*, 2014, pp. 844–848.
- [10] M. Grinberg, *Flask web development: developing web applications with python*. ” O’Reilly Media, Inc.”, 2018.
- [11] A. Bucko, K. Vishni, B. Krasniqi, and B. Rexha, “Enhancing JWT authentication and authorization in web applications based on user behavior history,” *Computers*, vol. 12, no. 4, p. 78, apr 2023.
- [12] M. A. Budiman, “A comprehensive introduction to tailwind css,” Sep 2023. [Online]. Available: <https://medium.com/@alifm2101/a-comprehensive-introduction-to-tailwind-css-36bc9cb81a1c>
- [13] J. D. Luca, “Shadcn/ui add components and resources,” Jul 2024. [Online]. Available: <https://medium.com/@jidefr/shadcn-ui-add-components-and-resources-0846b0f57596>
- [14] A. Goode, B. Gilbert, J. Harkes, D. Jukic, and M. Satyanarayanan, “Openslide: A vendor-neutral software foundation for digital pathology,” *Journal of pathology informatics*, vol. 4,

p. 27, 09 2013.

- [15] R. Wetteland, K. Engan, T. Eftestøl, V. Kvikstad, and E. Janssen, “A multiscale approach for whole-slide image segmentation of five tissue classes in urothelial carcinoma slides,” *Technology in Cancer Research & Treatment*, vol. 19, p. 153303382094678, 10 2020.
- [16] O. Bchir, M. M. Ben Ismail, and H. Aljam, “Region-based image retrieval using relevance feature weights,” *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 18, no. 1, p. 65–77, Mar. 2018. [Online]. Available: <http://dx.doi.org/10.5391/IJFIS.2018.18.1.65>