Isabela State University Linker:
Journal of Education, Social Sciences, and Allied Health
Volume 2, Issue 1
ISSN (Print): 3082-3692 ISSN (Online): 3082-3706
Publisher: Isabela State University, Desktop Publishing, Echague, Isabela

Real-Time CNN-Based Generic Medicine Name Classification for Prescription Recognition

Espinas, Era B.[1], **Galvez, Leika Anne P.**[2], **Merino, Ma. Yalaine S. J.**[3], **Ramos, John Michael C.**[4], **Soriano, Ronan D.**[5]
College of Informatics and Computing Studies, New Era University, Quezon City, 1107, Philippines [1,2,3,4,5]
ebespinas@neu.edu.ph; leikaanne.galvez@neu.edu.ph; ma.yalaine.merino@neu.edu.ph; johnmichael.ramos@neu.edu.ph; ronan.soriano@neu.edu.ph

RESEARCH ARTICLE INFORMATION

Received: April 16, 2025
Reviewed: May 18, 2025
Accepted: June 26, 2025
Published: June 30, 2025

ABSTRACT

Illegible handwriting in prescriptions remains a prevalent issue in fast-paced healthcare environments, contributing to misinterpretation and medication errors. Recognizing the importance of prescription readability, this study developed and deployed a real-time web application that classifies 20 generic medicine names using a Convolutional Neural Networks (CNN) model, making the system more accessible for real-world use. A dataset of 2,100 images was collected from public and private hospitals in Quezon City, and expanded to 6,720 images using data augmentation techniques such as brightening, blurring, and noise reduction. Some medicines, such as Chlorpromazine and Hydroxyzine, showed slightly lower performance, suggesting the need for more diverse data. The results demonstrate the model's reliability and potential for integration into hospital systems or pharmacy management software, offering a practical solution to reduce errors in medication dispensing. Future work could involve expanding the dataset and integrating the model with OCR or electronic health record (EHR) systems to support broader handwriting variations and real-time clinical workflows.

Keywords: *Computer vision, Convolutional Neural Networks (CNN), medicine name classification, AI in medicine, web application*

Introduction

The rapid evolution of technology continues to impact many aspects of modern life, including healthcare. Among these innovations, computer vision has become increasingly important due to its ability to process and interpret visual data. As Parashar (2017) highlighted, computer vision has significantly influenced various domains, demonstrating notable reliability and accuracy. One of its crucial applications is in the recognition of handwritten text, an area that extends beyond traditional character recognition systems, which mainly translate images or documents into searchable formats (Memon et al., 2020). Handwriting recognition further enables machines to interpret and process the natural variability of human writing, a task that remains a critical challenge within intelligent document processing systems.

A long-standing issue in healthcare is the legibility of handwritten medical prescriptions. Poor handwriting, particularly among doctors, has become a stereotype due to the demanding nature of their work. During peak hospital hours, doctors may write prescriptions hastily, leading to increased risks of medication errors. Ratanto et al. (2021) found that the workload of nurses significantly contributes to such errors, as they are often responsible for interpreting and administering prescriptions under pressure. Cerio et al. (2015) also emphasized that a majority of medication errors stem from the illegibility of doctors' handwriting. Furthermore, the similarity in spelling and appearance among many medicine names—often derived from complex chemical components—adds another layer of risk in prescription interpretation (Tabassum et al., 2022). These problems highlight a crucial need for reliable systems that can reduce interpretation errors and improve patient safety.

Handwriting recognition (HWR), also known as handwritten text recognition (HTR), refers to the capability of computational systems to interpret handwritten input from various sources like paper, photos, and touchscreens (Prasantha, 2023). By transcribing handwritten documents into digital text, HWR holds substantial promise for applications in healthcare. Machine learning, particularly deep learning, has long been applied to enhance the quality and accuracy of healthcare systems (Shehab et al., 2022). Kamble and Mane (2024) acknowledged the ongoing challenges in this domain, particularly the variability of individual handwriting styles—such as differences in character size, stroke, and form—which complicates standardization. As Shrawankar (2019) pointed out, the unstructured nature of human handwriting makes consistent recognition difficult, especially when cursive and print styles intermingle.

In response to these challenges, Convolutional Neural Networks (CNNs) have emerged as a powerful solution for image processing and classification tasks, including handwriting recognition. These networks simulate the visual processing capabilities of the human brain through layered structures that identify and categorize image patterns (Taye, 2023). Several studies have demonstrated the potential of CNNs for interpreting handwritten medical text. Jain et al. (2021) presented a pipeline combining CNNs with bi-directional LSTM networks to convert handwritten prescriptions into readable digital text, while Fajardo et al. (2019) achieved 76% training accuracy and 72% validation accuracy using a Deep Recurrent Convolutional Neural Network (Deep R-CNN) for similar tasks. These successes underscore the promise of CNN-based systems in minimizing medication errors and improving efficiency in medical documentation.

To translate these advancements into practical, user-friendly tools, this study proposed the deployment of the CNN model in a web-based application. Web applications, which operate through browsers and often use host-server architectures, are well-suited for real-time interaction and accessibility across platforms (OjaswiTech,

n.d.). Sotnik et al. (2023) emphasized the benefits of such platforms, including cross-device compatibility and ease of access without requiring software installation. Thokala (2021) also noted that machine learning integrated into web applications can enhance user experience and operational efficiency. The use of modern JavaScript libraries such as React JS and TensorFlow.js further supports rapid front-end development and enables direct execution of machine learning models within browsers (Smilkov et al., 2019). React JS simplifies the creation of dynamic and aesthetic interfaces (Bhalla et al., 2020), while TensorFlow.js facilitates real-time inference of models directly on the client-side.

Despite the growing number of studies utilizing CNNs for handwriting recognition and the increasing feasibility of web-based deployments, there remains a gap in systems specifically tailored to address prescription legibility in real-time, particularly those focused on generic medicine names. This study aimed to bridge that gap by designing, developing, and evaluating a CNN-based handwriting recognition model for integration into a web application. The system is designed to classify 20 generic medicine names and provide relevant descriptions and indications. The research encompasses dataset expansion through data collection and augmentation, model training, front-end integration, and evaluation using precision, recall, accuracy, and F1-score. By targeting a real-world problem with direct clinical implications, this study contributes a practical and innovative solution to medication safety in healthcare.

Methods

This study aimed to develop a system that can accurately identify handwritten generic medicine names from prescription samples in real time. The main objective was to create a tool that can help health practitioners quickly and accurately recognize medication names, thereby reducing medication errors and improving service delivery. To achieve this, a step-by-step approach was followed—beginning with data collection and preparation, followed by the development and training of a machine learning model, the design of a web application for real-world use, and finally, testing and deployment of the application.

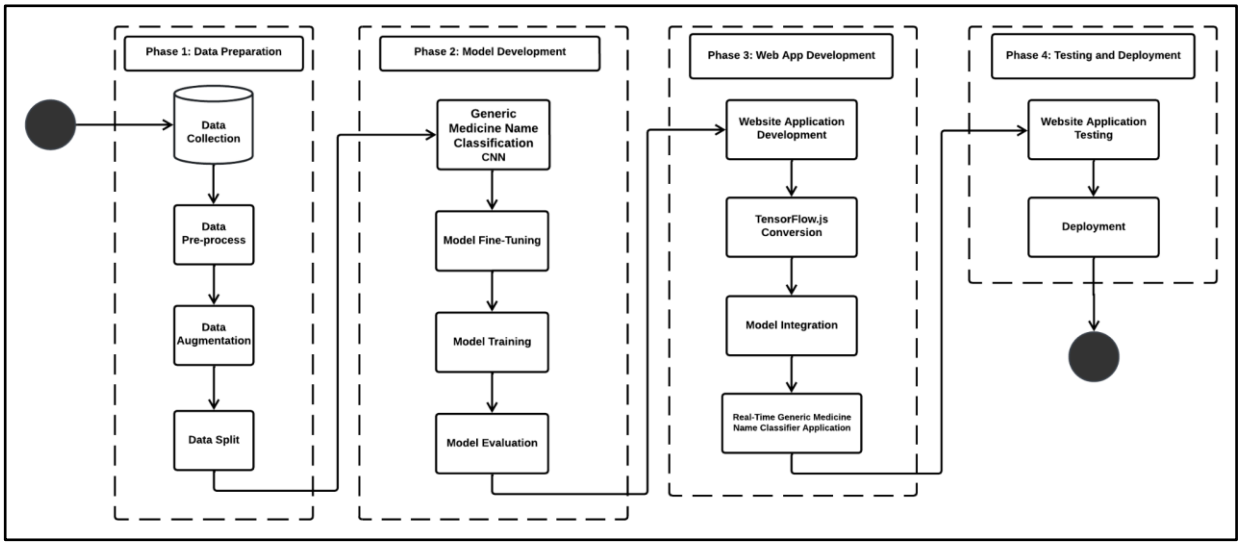


Figure 1. *Project Design for the Proposed System*

Figure 1 shows the project design for the proposed system, which was planned and followed by the researchers in conducting this study. The whole development process is divided into four key phases: data preparation, model development, web app development, and testing and deployment.

Installation of Tools and Libraries

To ensure reproducibility and technical clarity, this section details the tools and libraries used in the study, including their specific version, installation method, function, and role within the development pipeline. This structured documentation aligns with best practices in deep learning and computer vision research and supports the reproducibility of experimental results.

Albumentations

Albumentations (v1.4.0) is a fast and flexible image augmentation library designed to improve model generalization through a diverse set of transformations. It was used to apply operations such as brightness adjustment, Gaussian noise, motion blur, and flipping to augment the handwritten image dataset. This preprocessing step helps the model become more robust to visual variations in handwritten medicine names. It was installed using !pip install albumentations==1.4.0 and integrated within the data augmentation pipeline in Google Colab notebooks.

Google Colab Pro

Google Colab Pro provides an online hosted Jupyter Notebook environment with access to high-performance GPUs and TPUs, crucial for training computationally intensive deep learning models. No setup or local configuration is needed, and its seamless compatibility with Python-based libraries made it an ideal environment for developing the CNN-based classification system. It also allowed collaborative editing and persistent storage using Google Drive integration.

Google Drive

Google Drive was utilized as the primary cloud storage medium for handling datasets, storing trained model weights, and saving logs. It was mounted in the Colab environment using the command from Google.colab import drive, enabling read/write access to local project folders. This integration ensured that all project files were consistently backed up and accessible across sessions.

Jupyter Notebook

Jupyter Notebook (v6.5.4) served as the coding interface within Google Colab, providing a document-oriented environment for scripting, documenting, and executing Python code. It allowed for inline visualization of model metrics and supported rich text annotations, which facilitated experimentation and debugging during development.

Keras

Keras (v2.13.1) is a high-level neural network API built on top of TensorFlow and is used to define and train the convolutional neural network

architecture. It simplified model creation through intuitive APIs and abstracted much of the underlying complexity, thereby accelerating development. It was installed via !pip install keras==2.13.1 and used throughout the training and evaluation phases.

Matplotlib

Matplotlib (v3.7.1) is a plotting library used to visualize training curves, such as loss and accuracy per epoch. These plots provided insight into the learning dynamics of the model and helped in identifying issues like overfitting or underfitting. It was installed using !pip install matplotlib==3.7.1 and used in conjunction with Seaborn for enhanced visualizations.

Node.js

Node.js (v18.x LTS) was used to support TensorFlow.js model inference outside the browser, particularly in server-side environments. It allowed for deploying the trained CNN model into a lightweight and scalable backend, useful for applications requiring real-time inference. It was downloaded from the official website and configured as part of the deployment stack.

NumPy

NumPy (v1.24.3) is a core Python library for numerical computations, especially efficient array processing. It was used extensively in preprocessing steps such as image reshaping, normalization, and batch manipulation. It was installed via !pip install numpy==1.24.3 and integrated into both training and augmentation scripts.

OpenCV

OpenCV (v4.7.0.72), or the Open-Source Computer Vision Library, provided image processing capabilities essential for preprocessing handwritten images. Operations like grayscale conversion, thresholding, and bounding box extraction were performed using this library to prepare the data before feeding it into the CNN. It was installed using !pip install opencv-python==4.7.0.72.

Pandas

Pandas (v1.5.3) was used for handling structured data, particularly during dataset organization and metadata processing. It facilitated tasks like reading label information from CSV files and mapping class names to image files. It was installed via !pip install pandas==1.5.3 and used primarily in the data preparation phase.

Random

The random module is part of Python's standard library and was used to generate pseudo-random numbers for tasks such as shuffling datasets and applying stochastic augmentation operations. Its use ensured that each training session introduced slight variations in data ordering and transformations, aiding generalization.

React.js

React.js (v18.2.0) was employed to develop the front-end interface of the web application, allowing users to interact with the trained model. It enabled the

creation of a dynamic and responsive UI where users could upload handwritten images and receive real-time predictions. It was installed using npm install react@18.2.0 and integrated into the deployment pipeline alongside TensorFlow.js.

Seaborn

Seaborn (v0.12.2) is a statistical data visualization library built on top of Matplotlib. It was used to generate more refined and interpretable plots of performance metrics, aiding in comparative analysis and model evaluation. It was installed using !pip install seaborn==0.12.2 and often used in tandem with Matplotlib.

Shutil

Shutil is a standard Python library used for high-level file operations, including copying and organizing image datasets into appropriate directory structures. It was especially useful during data augmentation, where new images needed to be saved into specific folders. No installation was required.

TensorFlow

TensorFlow (v2.13.0) served as the primary deep learning framework, handling model construction, training, evaluation, and exporting. It offers a comprehensive ecosystem for machine learning and was compatible with both Keras and TensorFlow.js, streamlining the development-to-deployment process. Installed via !pip install tensorflow==2.13.0, it formed the backbone of the study's model development pipeline.

TensorFlow.js

TensorFlow.js (v4.15.0) is a JavaScript library that enables running trained machine learning models in a browser or server-side using Node.js. It allowed the CNN model to be deployed into a web-based environment, ensuring fast, private, and accessible inference without server dependency. It was installed using npm install @tensorflow/tfjs@4.15.0.

Tqdm

Tqdm (v4.66.1) is a Python utility library that provides real-time progress bars for loops and processes. It was used to visually monitor training epochs, data preprocessing, and augmentation tasks. Installed via !pip install tqdm==4.66.1, it improved transparency and debugging efficiency during long-running operations.

Phase 1: Data Preparation

The first phase of developing this project aim to collect a suitable dataset and prepare the collected data for use in the machine learning model training process. By achieving these objectives, the machine learning model will gain a solid foundation to build upon in the later phases of the project's development.

Table 1. List of the Classes Used in the Study (Generic Medicine Names)

Prescription	Medication
Prescription 1	Azathioprine
Prescription 2	Ceftriaxone
Prescription 3	Chlorpromazine
Prescription 4	Ciprofloxacin
Prescription 5	Clarithromycin
Prescription 6	Dobutamine
Prescription 7	Fluoxetine
Prescription 8	Hydrochlorothiazide
Prescription 9	Hydrocortisone
Prescription 10	Hydroxyzine
Prescription 11	Ibuprofen
Prescription 12	Levothyroxine
Prescription 13	Lorazepam
Prescription 14	Metronidazole
Prescription 15	Prednisolone
Prescription 16	Quinine
Prescription 17	Risperidone
Prescription 18	Rituximab
Prescription 19	Salbutamol
Prescription 20	Tramadol

To gather the necessary data, attending doctors, medical practitioners, and students were asked to write sample prescriptions with the medicine names listed in Table 1. The final dataset consisted of 2,100 labeled images, 105 images for each of the 20 classes, which were then organized into separate folders by class name. Before training, all images were resized to 224x224 pixels, converted to grayscale, and processed with techniques such as gamma correction, contrast adjustment, noise removal, and Gaussian blur to enhance the visibility of the handwriting. To account for the small size of the dataset and to help the model learn more robust features, extensive data augmentation techniques were applied. These techniques included rotating, zooming, shifting, adding color variations, and adding noise to the images. After augmentation, the total number of images in the dataset increased to 6,720.

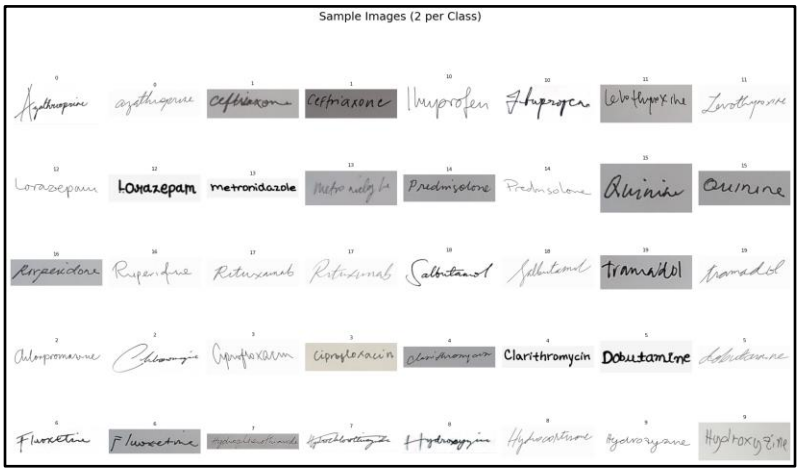


Figure 2. *Sample Images per Class Taken from the Dataset*

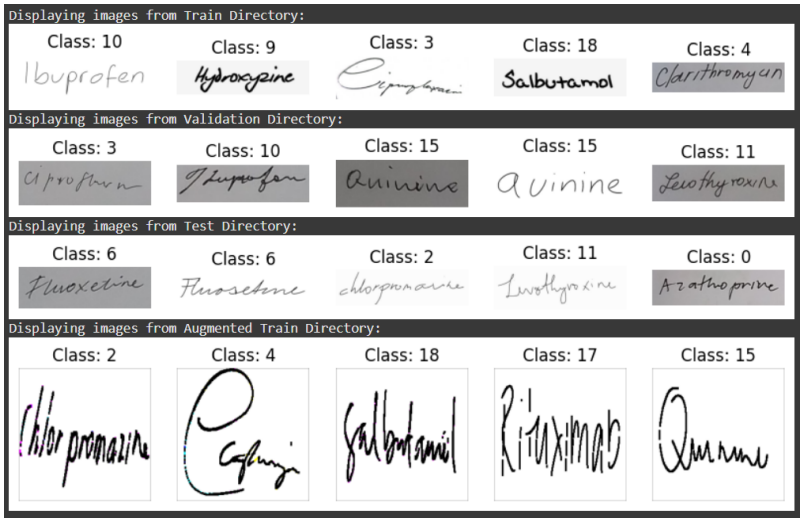


Figure 3. *Sample Images Taken from the Dataset After Splitting*

To avoid bias and imbalance, the dataset was split into training, validation, and testing sets in an 80:10:10 ratio, while retaining the proportions of each class. Furthermore, custom class weights were applied during training to help balance classes and avoid bias toward more frequently represented ones.

Once the data had been prepared and processed, the next phase involved developing and training a CNN that could accurately identify medication names from the images of handwritten prescriptions.

Phase 2: Model Development

This section of the study discusses how a CNN architecture was designed to create a generic medicine classification model, featuring 20 distinct classes from the dataset. For this study, a CNN architecture was implemented using the Keras API with a TensorFlow backend. The main objective was to create a classifier that could identify 20 classes of medication accurately. The architecture starts with a convolution layer with 32 filters and a kernel size of 3×3, employing the Rectified Linear Unit (ReLU) as the activation function to output only positively activated signals. Subsequently, a max

pooling layer reduces the dimensions of the feature maps while retaining the most important information.

After flattening these feature maps, a fully connected layer with 128 neurons was added, followed by a dropout layer to help avoid overfitting. The final layer utilizes a softmax activation to produce a probability distribution across all 20 classes.

The network was trained for 100 epochs with a batch size of 32, employing the Adam optimizer and categorical cross-entropy as the loss function. To avoid overfitting, early stopping was implemented, and the weights from the epoch with the best performance were preserved.

$$f(x) - \max(0,x)$$

Equation 1. Relu Formula

The Rectified Linear Unit (ReLU) serves as the activation function in all hidden layers except the output layer. It operates by evaluating each input value: negative values are converted to 0, while positive values remain unchanged. This mechanism facilitates learning by allowing only significant positive values to propagate through the model.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Equation 2. *Softmax Formula*

The softmax activation function is applied to the CNN model's final dense layer. This layer generates raw output logits for each of the 20 classes. The softmax function transforms these raw values into positive numbers and normalizes them to sum to one, effectively creating a probability distribution across the 20 classes. This normalization enables straightforward interpretation of outputs as the model's confidence in each class. In practice, the class with the highest probability is selected as the model's prediction, ensuring that outputs are meaningful and ready for comparison with true class labels during training.

$$CE = -log\left(\frac{e^{s_p}}{\Sigma_j^c e^{s_j}}\right)$$

Equation 3. *Categorical Cross-Entropy Loss Formula*

To train the CNN, the researchers used categorical cross-entropy as the loss function. In simple terms, for each sample, there is a true label indicating its correct class, and the model predicts a probability for each class. The loss function measures how far off the predicted probabilities are from the true labels.

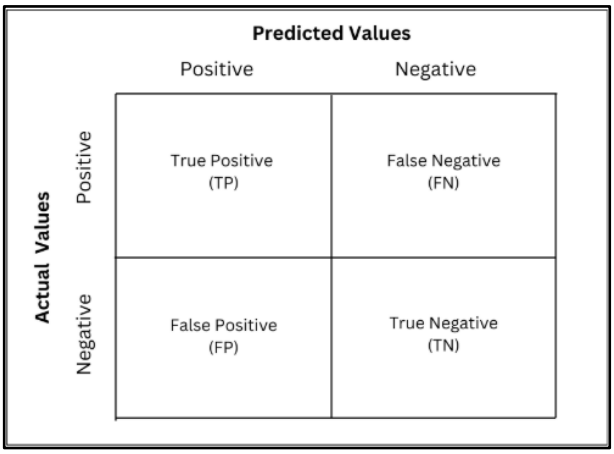


Figure 4. *Visualization of the Confusion Matrix*

After the training process of the model, it was then evaluated using four (4) key metrics utilizing a Confusion Matrix as seen in Figure 4. The accuracy, precision, recall, and F1-score were measured to determine whether the model was fit for usage.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Equation 4. *Formula for Precision*

Precision determines the percentage of predicted positive classes that were actually positive.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Equation 5. *Formula for Recall*

Recall, also known as True Positive Rate, measures the proportion of the actual positives that were correctly predicted by the classifier.

$$F1 - Score = \frac{2TP}{2TP + FN + FP}$$

Equation 6. *Formula for F1- Score*

The F1-score is a singular metric that combines both precision and recall. F1-score is usually used to obtain a balanced evaluation of both precision and recall.

$$Accuracy = \frac{TPn}{N}$$

Equation 7. *Formula for Accuracy*

Lastly, accuracy is the percentage of samples the classifier correctly identified.

Once the trained model demonstrated strong performance, the next step was to implement it in a web application. The following phase focuses on developing a graphical interface that allows users to scan and identify medication from handwritten prescriptions in real time.

Phase 3: Web Application Development

This phase involved designing and developing a user-friendly web application that can scan handwritten medication names in real time and instantly provide identification results.

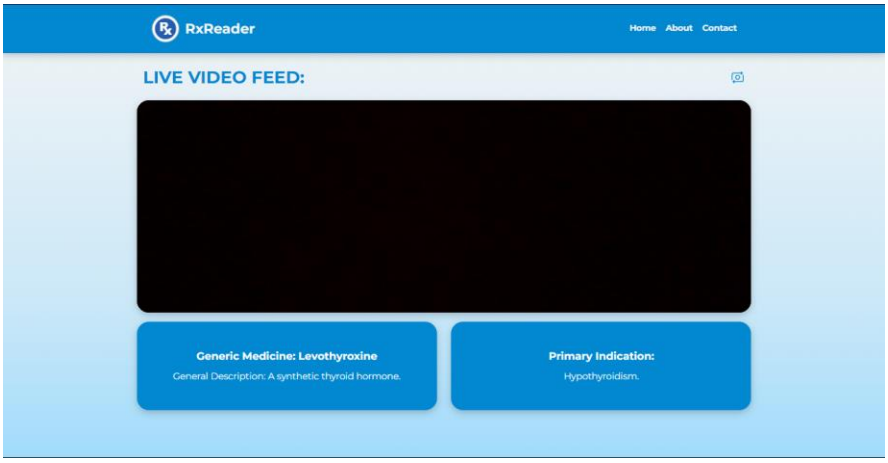


Figure 5. *Created UI of the Application for Desktop Devices*

Figure 6. *Created UI of the Application for Mobile Devices*

Figure 5 and Figure 6 both showcase the different types of devices that can use the application. Using React and TypeScript, a lightweight and responsive UI was implemented to run on both desktop and mobile devices. The application utilizes the WebRTC API to activate the phone's or computer's camera and capture video frames in real time.

Every second, a frame is resized to 224×224, converted into a tensor, normalized, and fed into the trained model. The frames captured by the device's camera need to be resized in order to fit the requirements of the model for processing. The application then displays the predicted medication's name, confidence score, a brief description, and its main use directly on the screen.

To maintain lightweight functionality and protect patient confidentiality, no data is stored or transmitted; all processing occurs directly in the browser. Furthermore, error handling mechanisms were implemented to account for cases where the camera fails or a medication is not recognized. Unit and integration testing were performed to validate the application's functionality across different devices and conditions.

With the application fully implemented, the final phase was testing and deployment. The following section describes how the application was evaluated to assess its accuracy and readiness for deployment.

Phase 4: Testing and Deployment

Phase 4 focused on testing and preparing the application for deployment. For this final phase, the application was thoroughly tested with various handwritten prescription samples to assess its accuracy and reliability. The web application was tested using various handwritten prescription samples to ensure it worked reliably and accurately. Basic security measures were implemented to align with health data standards (such as HIPAA), although the application does not store or send any patient information. The main purpose of the application is simply to scan, identify, and provide information about medication in real time.

Once testing was complete, the application was deployed on Railway, a reliable service for hosting web applications. To help users quickly become comfortable with its use, clear instructions and a brief disclaimer were included on the website upon entering.

Ethical Considerations

To maintain ethical standards and protect contributors' confidentiality, all participating doctors were informed about the study and consented to provide samples of their handwriting. None of their identifying information was kept. The handwritten prescriptions were used exclusively for training, validation, and testing. Furthermore, the application itself does not store or track any patient data; it performs all operations directly in the browser and discards the photo immediately after processing.

Results and Discussion

This chapter presents the results of the study's implementation and evaluation. To thoroughly assess the performance of the trained classifier and its application, various graphs, tables, and a confusion matrix were generated. Each of these elements is explained in detail to provide a clear understanding of the model's capabilities and limitations. Furthermore, this chapter aims to connect these results back to the research questions and goals, offering a more insightful view into how well the model performs in identifying handwritten medication names.

To help thoroughly examine, analyze, and interpret the results of this study, graphs, tables, and other visual elements have been included. Each of these figures is explained in detail to provide a clear understanding of the findings.

Plotted Visual Graphs of the Results

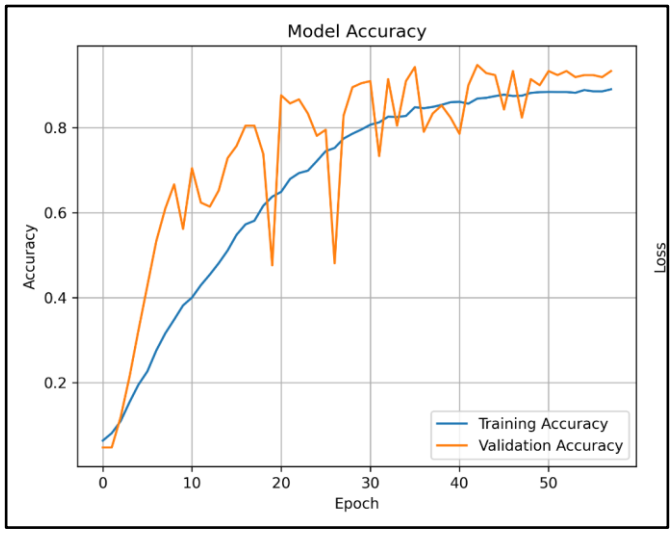


Figure 7. *Training and Validation Accuracy Graph*

As shown in Figure 7, the training accuracy (blue line) starts off low but steadily improves, becoming more stable around the 40th epoch and eventually reaching over 90%. This steady increase suggests that the model was learning effectively and making better predictions as training progressed. The validation accuracy (orange line) rises quickly within the first 10 epochs but shows some fluctuations afterward, indicating that the model may have struggled to generalize at certain points. However, after around the 50th epoch, the validation accuracy also stabilizes and stays consistently above 90%. Despite the earlier variability, the close alignment of both training and validation accuracy by the end of training suggests that the model not only learned well from the training data but also generalized successfully to unseen validation samples.

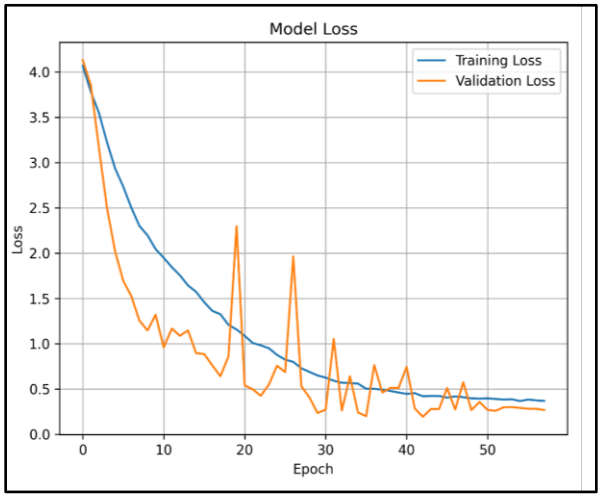


Figure 8. *Training and Validation Loss Graph*

Figure 8 displays the training and validation loss curves over the course of more than 50 epochs. The training loss (blue line) decreases steadily and smoothly, indicating that the model is effectively minimizing error during training. In contrast, the validation loss (orange line) shows noticeable spikes during the early stages, suggesting some initial instability, possibly due to variance in the validation set or challenges in generalizing early on. However, as training progresses, the validation loss also stabilizes and follows a similar downward trend. The relatively small gap between the training and validation loss toward the end of training indicates that the model is not overfitting and is learning to generalize well to unseen data. Overall, despite some early fluctuations, the model successfully reduced its prediction errors and produced more accurate results by the end of training.

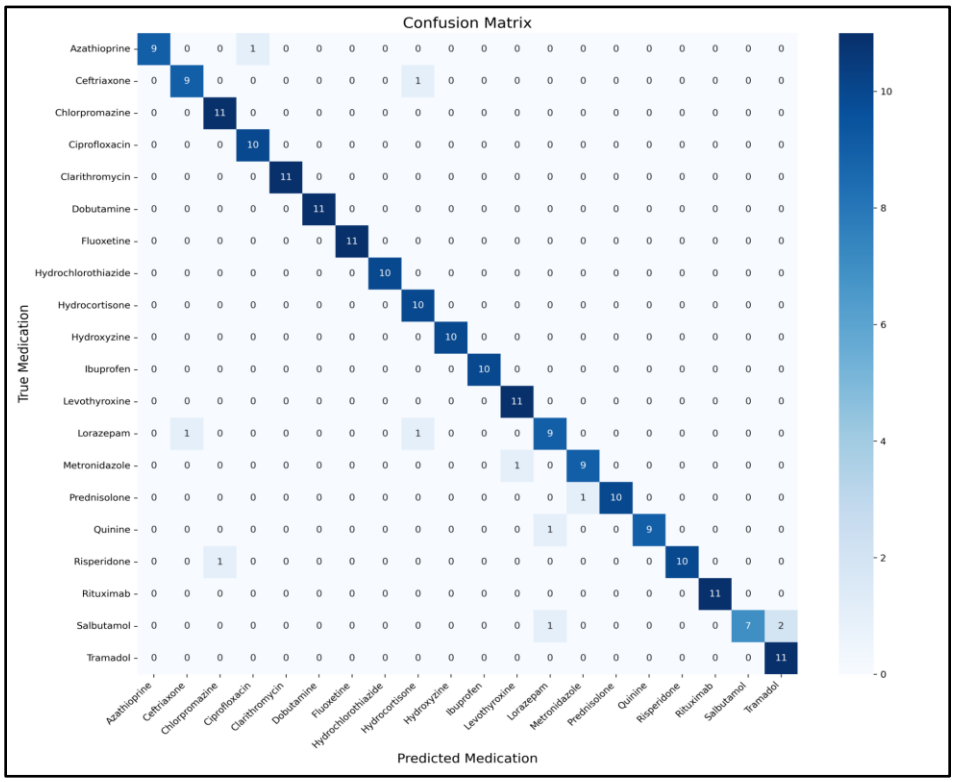


Figure 9. *Class Confusion Matrix*

To further evaluate the classifier's performance, a confusion matrix and a detailed classification report were generated. These metrics collectively reflect how accurately the model identifies each medication class and highlight areas where confusion occurs.

The confusion matrix in Figure 9 depicts the performance of a generic medicine name classification model across 20 different medications. The diagonal elements represent correct predictions, showing high accuracy for most medications with values between 9–11 correct classifications per medication. Notable observations include above-average classification for several medications, including Chlorpromazine, Clarithromycin, Dobutamine, Fluoxetine, Levothyroxine, Rituximab, and Tramadol. Some confusion occurs between specific medication pairs: Azathioprine is occasionally misclassified as Ciprofloxacin; Ceftriaxone and Lorazepam show mutual confusion with Hydrocortisone; and Salbutamol exhibits the highest misclassification rate, being confused with Tramadol in 2 cases. The lack of certainty between the mentioned classes

arises because some of these medicines have similar handwriting patterns or abbreviations, reflecting a challenging, domain-specific problem in medical handwriting recognition. The overall pattern, however, suggests strong model performance with limited cross-medication confusion.

Table 2. Model Classification Report

Prescription	precision	recall	f1-score	support
Azathioprine	1.00	0.90	0.95	10
Ceftriaxone	0.90	0.90	0.90	10
Chlorpromazine	0.82	0.82	0.82	11
Ciprofloxacin	0.90	0.90	0.90	10
Clarithromycin	1.00	0.91	0.95	11
Dobutamine	1.00	0.90	0.95	10
Fluoxetine	1.00	0.91	0.95	11
Hydrochlorothiazide	1.00	1.00	1.00	11
Hydrocortisone	1.00	0.70	0.82	10
Hydroxyzine	0.85	1.00	0.92	11
Ibuprofen	0.92	1.00	0.96	11
Levothyroxine	0.91	1.00	0.95	10
Lorazepam	1.00	1.00	1.00	11
Metronidazole	1.00	1.00	1.00	11
Prednisolone	1.00	1.00	1.00	11
Quinine	1.00	1.00	1.00	10
Risperidone	0.83	1.00	0.91	10
Rituximab	1.00	1.00	1.00	10
Salbutamol	1.00	1.00	1.00	10
Tramadol	0.92	1.00	0.96	11
accuracy			0.95	210
macro avg	0.95	0.95	0.95	210
weighted avg	0.95	0.95	0.95	210

Table 2 presents the precision, recall, F1-score, and support for each of the 20 generic medicine name classes. The classification model achieved strong overall performance, with an accuracy of 94.76% (199 correct predictions out of 210 samples), and both macro and weighted average F1-scores of 0.95. Several medication classes, including Hydrochlorothiazide, Lorazepam, Metronidazole, Prednisolone, Quinine, Rituximab, and Salbutamol, performed consistently well. Achieving high scores across all three evaluation metrics indicates the model's ability to reliably identify these medications.

On the other hand, Chlorpromazine and Hydrocortisone showed relatively lower performance. Chlorpromazine recorded a precision, recall, and F1-score of 0.82, while Hydrocortisone had a high precision (1.00) but lower recall (0.70), resulting in an F1-score of 0.82. These findings suggest that certain medications may be more susceptible to misclassification, potentially due to similar handwriting characteristics or overlapping visual features. Overall, the model demonstrates reliable and balanced performance across most medication classes, though there remains some room for improvement in distinguishing more challenging cases.

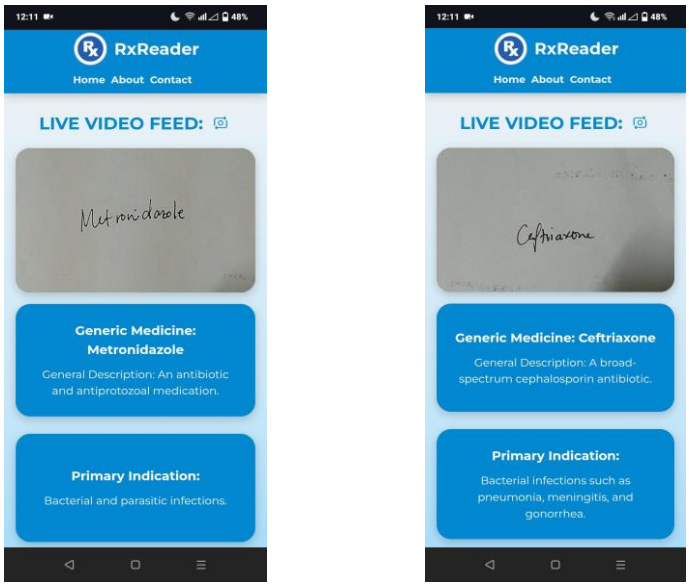
Table 3. Model Performance Summary

Model Performance Summary	
Test Accuracy:	0.9476
Test Loss:	0.2455
Total Test Samples:	210
Correct Predictions:	199 (94.76%)
Incorrect Predictions:	11 (5.24%)

Table 3 provides a concise overview of the model's performance on the test set. It includes key metrics such as test accuracy, loss value, and the breakdown of correct versus incorrect predictions. The relatively low-test loss value of 0.2455 indicates that the model maintained a consistent level of prediction confidence, while the distribution of results highlighted its effectiveness across the evaluated samples.

After analyzing the overall performance of the model through key metrics such as accuracy, precision, recall, and F1-score, the study now transitions to the next phase of evaluation, which involves the implementation of the trained classifier into a real-world web application. This step aims to demonstrate not only how well the model performs under test conditions, but also how effectively it can be applied in practice to aid health practitioners in identifying medication names directly from handwritten prescriptions. The application was designed to enable health practitioners to scan and identify medication names directly from handwritten prescriptions.

Web Application Results



(10a) **(10b)**

Figure 10. *Screenshot of the Deployed Application on a Mobile Device*

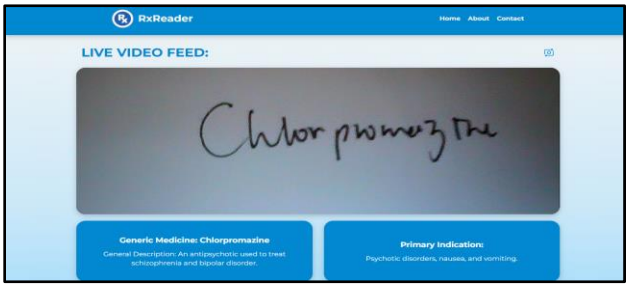
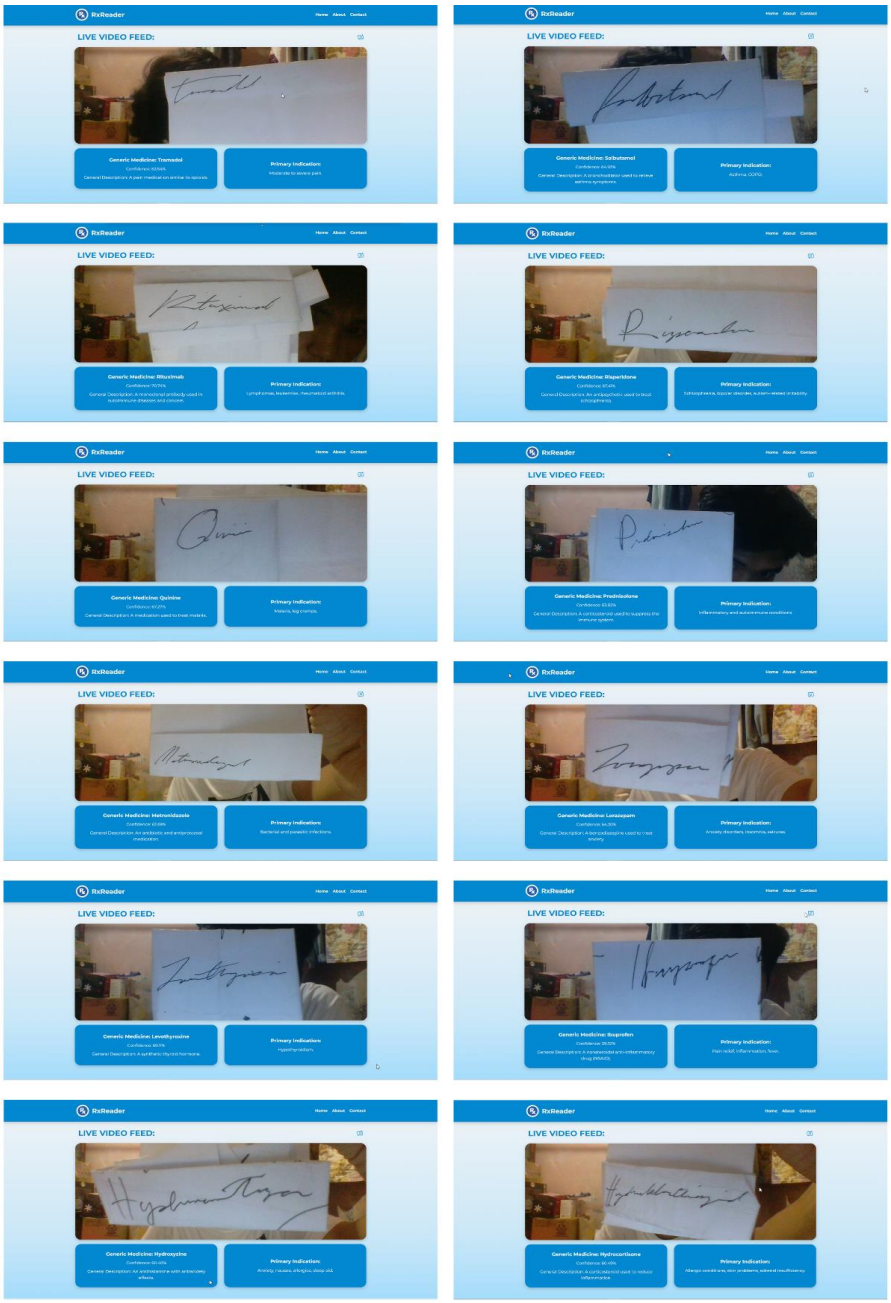


Figure 11. *Screenshot of the Deployed Application on a Desktop Device*



(12a)

(12b)

Figure 12. *Screenshots of Results per Class*

Figures 10 and 11 illustrate the deployed application in use on both mobile and desktop platforms. The app leverages the device's camera to capture a live video stream of handwritten prescriptions, allowing the trained classifier to perform real-time medication recognition. Upon detection, the application displays the predicted medication name along with its confidence score and primary use.

To assess its practical effectiveness, the application was tested across a variety of medication classes. As shown in Figure 12, it was able to identify multiple medications correctly in real time. However, for optimal performance, a steady hand, a clear image, and several environmental factors such as good lighting were often necessary to help the classifier make accurate predictions.

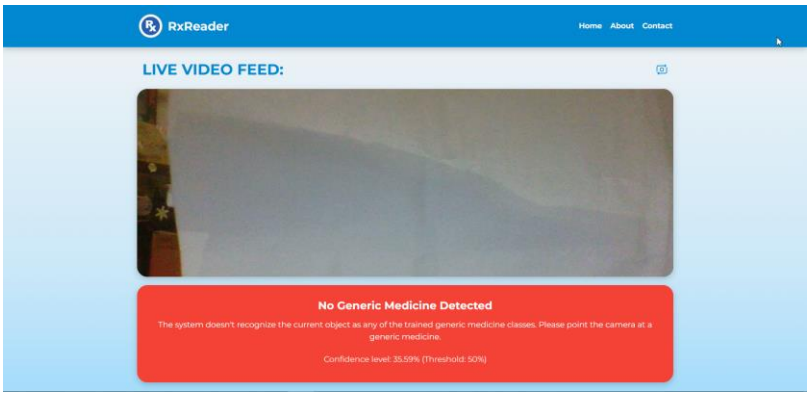


Figure 13. *Error Display with No Class Detected*

To account for cases where the classifier fails to match a medication, the application displays an error message as shown in Figure 13. This serves as a notification for the user to know that no medication was recognized.

Overall, the results demonstrate that the trained CNN performs strongly in identifying handwritten medication names from prescription samples, yielding a 94.76% accuracy and an F1-score of 0.95. The confusion matrix highlights areas of confusion, mainly between a few medication pairs, which may be addressed through additional training data or finer-tuned augmentation techniques.

Furthermore, the deployment of the trained classifier into a lightweight, real-time application underscores its practicality for health practitioners in a busy pharmacy setting. Nevertheless, the results also suggest areas for future improvement, particularly in addressing cases of confusion and improving its flexibility against poor handwriting and low-resolution images.

Conclusion and Future Works

This study addressed the persistent issue of illegible handwritten prescriptions, which often lead to medication errors in clinical settings. By developing a CNN-based system capable of recognizing 20 generic medicine names, the researchers successfully created and deployed a real-time web application using Google Colab and Railway Hosting Services. This lightweight browser-based tool allows users to accurately identify prescribed medications and access related information without storing sensitive data.

The CNN model achieved strong classification performance, showing its practical potential in real-world use. This solution contributes to improving medication safety by reducing the risk of misreading handwritten prescriptions. It also supports the ongoing shift toward digital healthcare, especially in the use of AI for document processing and real-time clinical tools.

The system helps healthcare professionals by improving accuracy in medication recognition, encouraging safer dispensing, and helping patients better understand their prescriptions. To improve its effectiveness, future work should expand the dataset to include more handwriting styles, integrate Optical Character Recognition (OCR) for more detailed text analysis, and optimize performance on mobile devices by using lighter models or compression techniques.

Testing this application in real healthcare settings like hospitals, clinics, or pharmacies is necessary to confirm its reliability and usefulness. Doing so could reduce medication-related errors and improve the way prescriptions are processed in everyday healthcare.

References

[1] Bhalla, A., Garg, S., & Singh, P. (2020). Present day web-development using ReactJS. *International Research Journal of Engineering and Technology (IRJET)*, *7*(5), 4410–4413. https://www.academia.edu/download/64560056/IRJET-V7I5223.pdf

[2] Cerio, A. A. P., Mallare, N. A. L. B., & Tolentino, R. M. S. (2015). Assessment of the legibility of the handwriting in medical prescriptions of doctors from public and private hospitals in Quezon City, Philippines. *Procedia Manufacturing*, *3*, 90–97. https://doi.org/10.1016/j.promfg.2015.07.112

[3] Fajardo, L. J., Sorillo, N. J., Garlit, J., Tomines, C. D., Abisado, M. B., Imperial, J. M. R., Rodriguez, R. L., & Fabito, B. S. (2019, November 1). Doctor's cursive handwriting recognition system using deep learning. In *2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)* (pp. 1-6). IEEE. https://doi.org/10.1109/hnicem48295.2019.9073521

[4] Jain, T., Sharma, R., & Malhotra, R. (2021). Handwriting recognition for medical prescriptions using a CNN-Bi-LSTM model. In *2021 6th International Conference on Inventive Computation Technologies (ICICT)* (pp. 824-829). IEEE. https://doi.org/10.1109/i2ct51068.2021.9418153

[5] Kamble, N. C. B., & Mane, N. K. T. (2024). A review on handwritten recognition system using machine learning techniques. *International Research Journal on Advanced Engineering Hub (IRJAEH)*, *2*(6), 1590–1599. https://doi.org/10.47392/irjaeh.2024.0218

[6] Memon, J., Sami, M., Khan, R. A., & Uddin, M. (2020). Handwritten optical character recognition (OCR): A comprehensive systematic literature review (SLR). *IEEE Access*, *8*, 142642–142668. https://doi.org/10.1109/access.2020.3012542

[7] OjaswiTech. (n.d.). *Web applications, their use, benefits and importance of web applications today*. OjaswiTech. Retrieved April 12, 2025, from https://www.ojaswitech.com/web-design-technologies/web-applications

[8] Parashar, A. (2017). Importance of computer vision for human life. *International Journal of Advanced Research*, *5*(3), 2396–2399. https://doi.org/10.21474/ijar01/3769

[9] Prasantha, S. (2023). Detailed survey of handwriting recognition using machine learning algorithms. *International Journal of Creative Research Thoughts*, *11*(1), B704–B708. https://www.researchgate.net/publication/377626296_Detailed_Survey_Of_Handwriting_Recognition_Using_Machine_Learning_Algorithms

[10] Ratanto, N., Hariyati, R. T. S., Mediawati, A. S., & Eryando, T. (2021). Workload as the most important influencing factor of medication errors by nurses. *The Open Nursing Journal*, *15*(1), 204–210. https://doi.org/10.2174/1874434602115010204

[11] Shehab, M., Abualigah, L., Shambour, Q., Abu-Hashem, M. A., Shambour, M. K. Y., Alsalibi, A. I., & Gandomi, A. H. (2022). Machine learning in medical applications: A review of state-of-the-art methods. *Computers in Biology and Medicine*, *145*, Article 105458. https://doi.org/10.1016/j.compbiomed.2022.105458

[12] Shrawankar, U. (2019). Standardization of handwritten words to improve readability. *International Journal of Technology Diffusion*, *10*(3), 1–17. https://doi.org/10.4018/ijtd.2019070101

[13] Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., Zhang, K., Cai, S., Nielsen, E., Soergel, D., Bileschi, S., Terry, M., Nicholson, C., Gupta, S. N., Sirajuddin, S., Sculley, D., Monga, R., Corrado, G., Viégas, F. B., & Wattenberg, M. (2019, January 16). *TensorFlow.js: Machine learning for the web and beyond*. arXiv. https://arxiv.org/abs/1901.05350

[14] Sotnik, S., Shakurova, T., & Lyashenko, V. (2023). Development features of web-applications. *International Journal of Academic and Applied Research (IJAAR)*, *7*(1), 79–85. https://openarchive.nure.ua/entities/publication/0e1b6221-a2c0-4f68-b550-53714391c127

[15] Tabassum, S., Abedin, N., Rahman, M. M., Rahman, M. M., Ahmed, M. T., Islam, R., & Ahmed, A. (2022). An online cursive handwritten medical words recognition system for busy doctors in developing countries for ensuring efficient healthcare service delivery. *Scientific Reports*, *12*(1), 3840. https://doi.org/10.1038/s41598-022-07571-z

[16] Taye, M. M. (2023). Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions. *Computation*, *11*(3), 52. https://doi.org/10.3390/computation11030052

[17] Thokala, V. S. (2021). Integrating machine learning into web applications for personalized content delivery using Python. *International Journal of Current Engineering and Technology*, *11*(6), 1083–1087. https://www.researchgate.net/profile/Vasudhar-Sai-Thokala-2/publication/386523063_Integrating_Machine_Learning_into_Web_Applications_for_Personalized_Content_Delivery_using_Python/links/67540547ad10b614ef361ee9/Integrating-Machine-Learning-into-Web-Applications-for-Personalized-Content-Delivery-using-Python.pdf

Acknowledgment

The authors would like to express their sincere appreciation to all those who supported and contributed to the successful completion of this study.

To their families, the authors are truly grateful for the constant love, encouragement, and patience throughout this journey. To their friends, they extend heartfelt thanks for their understanding, support, and help during challenging phases of the research.

The authors also thank the professors of New Era University for their valuable guidance and expertise, which greatly shaped this study. They are sincerely appreciative of the doctors, medical practitioners, and students who provided handwritten prescription samples, making this research possible.

Lastly, the authors are thankful to the Almighty God for the blessings, strength, and direction that made the completion of this project a reality.

Conflict of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Artificial Intelligence (AI) Declaration Statement

Researchers hereby disclose the extent of the use of artificial intelligence (AI) tools, specifically ChatGPT, during the writing of the manuscript; this tool was used solely to check grammatical errors and improve grammatical structure.