VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JnanaSangama, Belagavi– 590018

Mini Project Report

On

"HANDWRITTEN PRESCRIPTION RECOGNITION:CNN & OCR"

Submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE & ENGINEERING

By

ROHAN A S	**4MT22CS132**
SAMRUDHI	**4MT22CS142**
SINCHANA	**4MT22CS159**
VARUN S	**4MT22CS180**

Under the Guidance of

Mr. Shreejith K B

Senior Assistant Professor

MITE

Invent Solutions

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MANGALORE INSTITUTE OF TECHNOLOGY &ENGINEERING

BadagaMijar, Moodabidri-574225, Karnataka

2024-2025

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

(A Unit of Rajalaxmi Education Trust ®, Mangalore)

Autonomous Institute Affiliated to V.T.U, Belagavi , Approved by AICTE, New Delhi Accredited by NAAC with A+ Grade & ISO 9001:2015 Certified Institution

MITE

Invent Solutions

DEPARTMENT OF COMPUTERSCIENCE & ENGINEERING

CERTIFICATE

This is to certify that Mr. **Rohan A S (4MT22CS132),** Ms**. Samrudhi (4MT22CS142),** Ms. **Sinchana (4MT22CS159),** Mr. **Varun S (4MT22CS180)** has satisfactorily completed the mini project entitled "**HANDWRITTEN PRESCRIPTION RECOGNITION:CNN & OCR"**for the **Mini Project (BCS586**) as prescribed by the VTU for 5th semester B.E. Computer Science & Engineering branch for the academic year 2024 – 2025. It is certified that all the corrections or suggestions indicated for internal assessment have been incorporated in the report and is been verified and validated.

……	**……**
Mr. Shreejith K B	**Dr. Ravinarayana B**
Mini Project Guide	**Head of the Department**

Name of Examiners	**Signature of the Examiners**
1. …….	**……..**
2. …….	**……..**

ABSTRACT

Doctors' hurried handwriting often leads to unclear prescription labels, causing mispronunciations and errors in identifying medications. To address this, a machine learning-based mobile application is proposed. The system preprocesses prescription images using noise reduction, resizing, and background removal to enhance clarity.

A Convolutional Neural Network (CNN) extracts features, helping generalize ***varied*** handwriting styles. The processed image is then analyzed using Optical Character Recognition (OCR), which identifies medication names. The results are matched against a medicine database, ensuring accuracy and providing detailed drug information, including usage and side effects.

This mobile app offers a user-friendly solution to improve prescription recognition and medication awareness.

ACKNOWLEDGEMENTS

The satisfaction and the successful completion of this project would be incomplete without the mention of the people who made it possible, whose constant guidance encouragement crowned our efforts with success.

This project is made under the guidance of **Shreejith K B**, **Senior Assistant Professor**, in the Department of Computer Science and Engineering. We would like to express my sincere gratitude to our guide for all the helping hand and guidance in this project.

We would like to express appreciation to **Dr. Ravinarayana B, Professor and Head**, Department of Computer Science and Engineering, for his support and guidance.

We would like to thank our Principal **Dr. Prashanth C M**, for encouraging us and giving us an opportunity to accomplish the project.

We also thank our management who helped us directly or indirectly in the completion of this mini project.

Our special thanks to faculty members and others for their constant help and support.

Above all, we extend our sincere gratitude to our parents and friends for their constant encouragement with moral support.

ROHAN A S **4MT22CS132**

SAMRUDHI **4MT22CS142**

SINCHANA **4MT22CS159**

VARUN S **4MT22CS180**

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Handwritten medical prescriptions are often difficult to interpret, leading to significant health risks. According to the National Academy of Science, over 1.5 million deaths or illnesses annually are linked to errors caused by illegible prescriptions or pharmacists' inability to identify medications correctly. The problem is exacerbated by the use of Latin abbreviations and medical jargon, making prescriptions challenging for individuals without medical training to comprehend. Misinterpreted prescriptions can result in severe health complications, particularly when incorrect drugs are consumed over extended periods. Additionally, many patients neglect to research potential side effects, increasing the likelihood of adverse reactions.

To address these issues, this study proposes a mobile application capable of accurately reading and interpreting handwritten prescriptions. Leveraging advanced technologies like Convolutional Neural Networks (CNN) and Optical Character Recognition (OCR), the application converts handwritten prescriptions into clear, digital text, identifying medication names and dosages. This system is designed to be user-friendly, making it accessible for both patients and pharmacists.A survey conducted among users and pharmacists revealed that 96% found the application highly beneficial, emphasizing its potential to transform prescription interpretation. This solution is particularly relevant in countries like Egypt, where handwritten prescriptions are prevalent, and there is a high demand for tools to enhance medication safety.

By reducing errors related to illegible prescriptions, this application offers a significant breakthrough in healthcare. It aims to improve patient outcomes by providing accurate drug information, reducing adverse reactions, and increasing awareness about prescribed medications. This innovation has the potential to save lives and enhance the overall safety of prescription handling worldwide.

1.2 Problem Statement

Handwritten medical prescriptions are often illegible, leading to misinterpretation of drug names, incorrect medication usage, and adverse health outcomes. This issue is exacerbated by medical jargon and patient unawareness of side effects. A solution is needed to digitize

and accurately interpret prescriptions, reducing errors and improving medication safety and accessibility.

1.3 Objectives

The Handwritten Prescription Recognition using CNN & OCR project sets out with three key objectives aimed at transforming the interpretation and digitization of handwritten medical prescriptions, addressing critical challenges in healthcare communication and medication safety.

- Accurate Recognition of Handwritten Prescriptions Using machine learning and TesseractOCR, the system will extract text from handwritten prescriptions. It will preprocess images to improve clarity and apply deep learning models to accurately recognize various handwriting styles, abbreviations, and distortions, ensuring reliable conversion of handwritten text to a machine-readable format.
- Identification and Matching of Medicines where the system will process recognized text with NLP techniques to extract drug names, dosages, and instructions. It will cross reference this data with a pharmaceutical database, ensuring accurate identification of medications, including both brand and generic names, while handling variations, misspellings, and abbreviations with fuzzy matching algorithms.
- Detailed Medicine Information , After identifying medications, the system will provide detailed information, including usage, dosages, side effects, drug interactions, and safety warnings. The data will be sourced from reliable medical databases and presented in an accessible, user-friendly format, ensuring informed decisions for patients, pharmacists, and healthcare providers.

1.4 Scope of the Project

The project aims to create an easy-to-use system that digitizes handwritten medical prescriptions using Optical Character Recognition (OCR) and Convolutional Neural Networks (CNN). It ensures accurate recognition of handwriting styles and matches medication names with a pharmaceutical dataset, identifying both brand and generic drugs. This system improves prescription accuracy, reduces errors, and streamlines healthcare processes for patients and pharmacists.

CHAPTER 2

LITERATURE SURVEY

A literature survey or a literature review in a project report is that section which shows the various analysis and research made in the field of our interest and the results already published, taking into account the various parameters and the extent of the project.

2.1 Existing System

[1] Esraa Hassan, HabibaTarek, Mai Hazem, ShazaBahnacy, LobnaShaheen, Walaa H. Elashmwai Faculty of Computer Science Misr International University, Cairo, Egypt esraa1707548, habiba1700246, mai1711462, shaza1708233, lobna.mostafa, walaa.hassan

Handwritten recognition, particularly in the context of doctors' handwriting, is a significant area of research due to its practical implications in healthcare. Several techniques have emerged to address this challenge, ranging from traditional Optical Character Recognition (OCR) to more advanced approaches like Convolutional Neural Networks (CNN) and hybrid algorithms.

[2] Sandhya P1 , Rama Prabha K.P2 , Jayanthi.R3 , V. Sujatha4 , Asha N5 , M B Benjulaanbu malar6 1Associate Professor, School of Computer Science and Engineering, VIT, Chennai 600127, Tamil Nadu, India sandhya.p@vit.ac.in Orcid:

0000-0002-0078-180X 2Assistant Professor Senior, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai 600127

The paper presentsFor example, the work by Kamalanaban et al. (2018) utilized a smartphone application combined with a medicine box and a CNN to accurately identify and interpret handwritten medicine names. Their findings indicated that increasing the volume of training data significantly improved the model's accuracy, highlighting the importance of robust datasets in enhancing recognition performance..

[3] Swasthika1 , Rithika C Jain2 , Sharanya3 , Shubhashree4 , AnnappaSwamy D R5 UG Student, Mangalore Institute of Technology and Engineering, Moodabidri,

Karnataka, India1,2,3,4 Associate Professor, Department of CSE, Mangalore Institute of Technology and Engineering, Moodabidri, Karnataka, India5

In addition to CNNs, other methods, such as Recurrent Neural Networks (RNNs) and hybrid approaches like the Whale Optimization Algorithm paired with neighborhood rough sets, have been explored. These methods aim to capture the sequential nature of handwriting and improve the overall interpretation of difficult-to-read text.

4] Maheen Islam1, Mahamudul Hasan1 and Nishat Vasker1 1Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh

Received Mon. 20, Revised Mon. 20, Accepted Mon. 20, Published Mon. 20

Researchers continue to investigate various strategies to enhance the accuracy and reliability of handwritten recognition systems, particularly for medical applications where misinterpretation can lead to serious consequences. Overall, the integration of machine learning techniques with real-world applications is paving the way for more effective solutions in this critical field.

2.2 Limitations of the Existing System

Despite significant advancements in handwritten recognition, particularly for doctor's handwriting, several challenges persist in the field. Traditional Optical Character Recognition (OCR) struggles with the variability in handwriting styles and the use of medical jargon and abbreviations, leading to frequent misinterpretations. Even more advanced approaches, like Convolutional Neural Networks (CNN), have limitations in handling the wide range of handwriting styles found in medical prescriptions, especially when trained on insufficient or unbalanced datasets.While methods such as Kamalanaban et al.'s smartphone application combined with CNNs have shown promise, their performance is heavily dependent on the quality and volume of training data. Small or nonrepresentative datasets can limit the system's ability to generalize, resulting in inaccurate recognition for poorly written or non-standard handwriting.

Additionally, while hybrid approaches like Recurrent Neural Networks (RNNs) and Whale Optimization Algorithm-based methods have been explored, they face challenges in scalability, computational complexity, and handling highly ambiguous or incomplete handwriting. Real-world applications of these systems require constant refinement to adapt

to different handwriting nuances, with ongoing research needed to ensure reliability in healthcare settings where errors can have severe consequences.

Thus, while existing systems are promising, their limitations in accuracy, data requirements, and handling complex, real-world handwriting hinder their full integration into medical practice.

2.3 Proposed System

The proposed system aims to address the limitations of existing handwritten prescription recognition technologies by incorporating advanced machine learning algorithms, including Convolutional Neural Networks (CNNs) and Optical Character Recognition (OCR) techniques. This integrated approach will enhance the accuracy and reliability of recognizing and digitizing medical prescriptions, especially those with challenging handwriting, typical of doctors' prescriptions.

The system will begin by utilizing OCR for basic text extraction, followed by CNNs to interpret complex handwriting patterns and nuances. The CNN model will be trained on a large, diverse dataset of medical prescriptions to improve its ability to recognize not only generic and brand names but also medical abbreviations, dosages, and instructions. This will ensure higher accuracy in the digitization process, even with illegible or poorly written text. To further enhance recognition, the system will use a hybrid approach, combining CNNs with Recurrent Neural Networks (RNNs) to capture the sequential nature of handwriting and improve recognition of connected or distorted characters.

Additionally, the system will include a robust database of pharmaceutical information to match recognized medicine names with their correct identities, providing users with detailed information on drug usage, dosages, and potential side effects. This will be complemented by an intuitive user interface that makes the system accessible to both healthcare professionals and patients, ensuring clarity and ease of use.

By combining these advanced techniques and ensuring the system can learn from diverse handwriting styles and complex data, this proposed system will improve the accuracy of handwritten prescription recognition and reduce medication errors, ultimately enhancing patient safety in healthcare environments.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

System Requirement Specification (SRS) serves as the foundational document outlining the detailed functional and non-functional requirements of a system, encompassing both software and hardware components. It delineates the system's purpose, scope, and functionalities, along with the constraints and performance parameters that must be adhered to during development. SRS acts as a blueprint for the entire development lifecycle, guiding stakeholders, developers, and testers in understanding and fulfilling the system's objectives. It facilitates effective communication among project stakeholders, ensuring a common understanding of the system's functionalities, performance expectations, and constraints, thus serving as a crucial reference throughout the development process.

3.1 SRS description

The SRS is a document, which describes completely the external behaviour of the software. This section of the SRS describes the general factors that affect the product and its requirements. The system will be explained in its context to show how the system interacts with other systems and introduce its basic functionalities.

3.1.1 Product Perspective

The proposed handwritten prescription recognition system is designed to address the ongoing challenges of accurately interpreting medical prescriptions, particularly doctors' handwriting, which has been a significant area of research. By leveraging advanced technologies like machine learning (CNNs) and Optical Character Recognition (OCR), the system will improve the clarity and accuracy of prescription data. Previous studies, such as those byMaheen Islam1, show that increasing training data and using hybrid algorithms can significantly enhance recognition accuracy. The system will integrate these insights into a robust application for both healthcare professionals and patients, ensuring seamless prescription digitization and reducing medication errors.

This product will offer a practical, real-world solution to improve patient safety by enhancing prescription recognition and providing comprehensive drug information, aiming to bridge the gap between handwritten prescriptions and reliable, actionable data.

3.1.2 Product Function

The proposed system will accurately recognize and digitize handwritten medical prescriptions using machine learning algorithms like CNN and OCR. It will extract medication names, dosages, and instructions from the text, matching them with a pharmaceutical database for accurate identification. The system will also provide detailed information about each medication, including usage, side effects, and dosage instructions, ensuring clarity, improving safety, and reducing medication errors for both healthcare providers and patients.

3.2 Specific Requirement

This section includes the detailed description about the hardware requirements, software requirements, functional requirements and non-functional requirements.

3.2.1 Hardware Requirement

Computer (PC or Laptop): For processing the images and running the model.

Storage Device: To save the database and any results from the processing.

CPU: A powerful processor or graphics card may be needed for faster processing if the model is computationally intensive.

Computer (PC or Laptop)

The hardware requirements for the Computer (PC or Laptop) in the proposed system focus on processing the images and running machine learning models. The computer or laptop should be equipped with a high-resolution display to visualize the processed data and code. A multi-core CPU or a powerful GPU is necessary to handle image processing tasks efficiently, especially when using resource-intensive models like Convolutional Neural Networks (CNNs). Additionally, a sufficient amount of storage is required to store large datasets, processed results, and the pharmaceutical database. The overall setup should facilitate seamless operation for recognizing and analyzing handwritten prescriptions in real-time.

Storage device

The hardware setup for the handwritten prescription recognition system includes essential storage devices to handle datasets and processed results. A camera phone is used to capture images of handwritten prescriptions, and a computer (PC or laptop) processes the images and runs the Optical Character Recognition (OCR) and Convolutional Neural Network (CNN) models. External storage devices, such as hard drives or SSDs, store the pharmaceutical database, prescription images, and results. The project uses a CPU for processing tasks, as GPU acceleration has not been utilized. This setup ensures that the system can process prescription images and identify medications accurately within the available hardware constraints.

CPU

For our handwritten prescription recognition project, a modern computer with a powerful CPU (central processing unit) is sufficient to handle the computational tasks effectively. The CPU manages key operations such as running the Flask server, processing images, and performing Optical Character Recognition (OCR) using Tesseract. Since the project utilizes a pre-trained model (VGG16) and does not involve training models from scratch, the use of a GPU (graphics processing unit) is not necessary. This setup ensures efficient performance without the need for high-end hardware like GPUs, making the project feasible with a standard computing device.

3.2.2 Software Requirement

Software requirements a field within Software of stakeholders that are to be solved by the software. The software requirements of our project given below:

Operating system

The choice of the operating system plays a crucial role in implementing a project like Handwritten Prescription Recognition using Convolutional Neural Networks (CNN) and Optical Character Recognition (OCR). Windows 10 or higher provides a robust and userfriendly platform for such projects, offering compatibility with a wide range of software and development tools.

The operating system's native support for OCR tools like Tesseract or integration with third-party OCR software enhances the system's ability to extract textual information from images. This feature is vital for processing and converting the handwritten prescription data into machine-readable formats. Windows' graphical user interface (GUI) simplifies application development, allowing developers to create user-friendly interfaces for healthcare professionals. Furthermore, the system's efficient memory management and multitasking capabilities enable the seamless execution of computationally intensive tasks like image pre-processing, feature extraction, and real-time recognition.

In summary, Windows 10 or later offers an excellent foundation for developing and deploying a Handwritten Prescription Recognition system, providing the necessary tools, compatibility, and performance required for success.

Python

Python is the backbone of the Handwritten Prescription Recognition system, serving as the primary programming language. Known for its simplicity and readability, Python enables developers to write clean and concise code, which is essential for implementing complex systems. It provides seamless integration with essential libraries and frameworks such as TensorFlow, OpenCV, and Pytesseract, making it a preferred choice for machine learning, deep learning, and image processing tasks. Python also supports a wide array of data manipulation libraries like NumPy and Pandas, which are useful for preprocessing and analyzing prescription data. Its strong community and extensive documentation ensure that developers have access to resources for troubleshooting and expanding their systems.

Python's compatibility across various platforms ensures that the code can run efficiently on Windows, Linux, or macOS, offering flexibility in deployment.

Pytesseract

Pytesseract, a Python wrapper for the Tesseract OCR engine, is employed to convert the textual content of prescription images into machine-readable text. It plays a vital role in extracting handwritten text, such as medicine names, dosages, and instructions. Pytesseract supports multiple languages and character sets, making it versatile for diverse handwriting styles and regional scripts. Its pre-trained capabilities for optical character recognition enable quick implementation without the need for extensive training data. Developers can integrate Pytesseract easily with image processing tools like OpenCV, creating a

streamlined workflow for text extraction. The extracted data can then be processed, validated, and stored for further use in the system. Pytesseract's accuracy improves when combined with image pre-processing techniques, ensuring reliable results.

Dataset

The dataset contains detailed information about various medicines, including their name, composition, uses, side effects, image URL, manufacturer, and user review percentages. The Medicine Name column identifies the product, while the Composition column specifies the active ingredients and their respective dosages. The Uses column outlines the medical conditions each medicine is prescribed for, such as cancer treatment, bacterial infections, and gastrointestinal issues. The Side Effects column lists potential adverse reactions, helping users be aware of the risks associated with each medicine. The Image URL provides a link to a visual representation of the medicine, aiding in product identification. The Manufacturer field indicates the company responsible for producing the medicine, ensuring authenticity. The Excellent Review %, Average Review %, and Poor Review % columns offer a breakdown of user feedback, giving insights into the overall satisfaction with each product. This dataset can be utilized for matching and retrieving relevant medicine information from prescription images.

React.Js

The React.js portion of the prescription recognition app provides a dynamic UI, efficiently managing states with use State. It handles file uploads via a component that captures images and sends them to the Flask backend using axios. Conditional rendering toggles between the welcome screen and the main interface. API integration enables asynchronous HTTP requests for OCR processing, dynamically displaying results. The app features a responsive layout with styled cards for readability and follows a component-based architecture for modularity and reusability.

VGG16

The updated project integrates the VGG16 model for image preprocessing before performing OCR with Tesseract. The image is resized to 224x224 pixels, converted to RGB if necessary, and passed through the pre-trained VGG16 model to extract feature maps. These CNN features can be utilized for advanced tasks like image classification or filtering. However, Tesseract remains responsible for extracting text, which is matched with medicine details from the CSV file.

Flask-CORS

In this code, Flask-CORS is used to enable Cross-Origin Resource Sharing (CORS), which is critical for allowing the frontend React.js application (running on a different origin, e.g., http://localhost:3000) to communicate with the backend Flask API. Without Flask-CORS, browsers would block such requests due to the Same-Origin Policy, which restricts web applications from accessing resources from another domain for security reasons The line CORS integrates Flask-CORS with the Flask app, enabling it to handle CORS requests automatically. This setup ensures that the React.js frontend can send requests (e.g., file uploads) to the Flask server and receive responses without being blocked by the browser Flask-CORS simplifies managing cross-origin headers and avoids the need for manual configuration, especially useful during development when frontend and backend often run on different ports. This integration ensures seamless communication in the prescription recognition application.

Flask

Flask is a lightweight and versatile web framework for Python, renowned for its simplicity and ease of use. Created by Armin Ronacher in 2010, Flask follows a minimalist approach, providing essential features for web development without unnecessary complexity. It offers intuitive routing, seamless integration with Jinja2 templating for dynamic content generation, and straightforward handling of HTTP requests and responses. Flask's modular architecture allows for easy extension with Flask extensions and third-party libraries, making it adaptable to various project requirements. With its built-in development server and reliance on Werkzeug and Jinja2, Flask empowers developers to quickly build web applications and APIs with Python.

Pillow

In this code, the Pillow library (a popular Python Imaging Library) is used for handling image files uploaded by users. Specifically, the Image module from Pillow is employed to open and process the uploaded image file before passing it to Tesseract for Optical Character Recognition (OCR).The line image = Image.open(file_path) loads the uploaded image from the temporary file storage. Pillow ensures compatibility with various image formats, such as JPEG, PNG, and BMP, allowing the application to process diverse image types. After loading, the image is passed to pytesseract.image_to_string(image) for text extraction.Pillow's ability to open, manipulate, and validate image files is crucial for

ensuring the uploaded file is an image and for preparing it for OCR. By integrating Pillow, the application becomes robust in handling different image formats and can seamlessly process prescription images for text extraction.

Pandas

In this code, the Pandas library is used to manage and manipulate the medicine details stored in a CSV file (medicine_details.csv). This CSV file likely contains information such as medicine names, composition, uses, manufacturers, and image URLs. The line medicine_data = pd.read_csv(csv_path) loads the data into a Pandas DataFrame, a versatile data structure that facilitates efficient querying and manipulation. Pandas is used here to iterate through the rows of the DataFrame (for _, row in medicine_data.iterrows()), matching the extracted text from the image with the corresponding medicine data. The matching logic compares the first five characters of the extracted text to the medicine names in the DataFrame. This integration with Pandas ensures efficient handling of structured data, enabling fast lookups, filtering, and conversion of rows into dictionaries for JSON responses. Pandas' capabilities make the app scalable for large datasets while simplifying the interaction with CSV-based medicine records.

TensorFlow

TensorFlow is used for processing the uploaded images through a pre-trained **VGG16 Convolutional Neural Network (CNN)** model. The VGG16 model, a deep learning architecture designed for image feature extraction, helps preprocess the prescription images by resizing and normalizing them to prepare them for Optical Character Recognition (OCR). This preprocessing step ensures that features extracted from the image are consistent and enhanced, improving the accuracy of downstream tasks, such as text recognition using Tesseract. This integration supports the objective of converting handwritten prescriptions into structured text data efficiently.

3.2.3 Functional Requirement

Functional requirements specify the essential operations, behaviors, and capabilities that a system must perform to meet its objectives. They describe what the system should do, focusing on its features and interactions. Functional requirements are critical for guiding system design and ensuring the end product meets user needs.

Image Input and Preprocessing: The system must allow users to upload prescription images in commonly used formats such as JPEG, PNG, or PDF while ensuring the proper preprocessing of these images, including resizing, noise removal, grayscale conversion, and contrast enhancement, to optimize them for recognition. Once the image is prepared, the system should detect and locate regions containing handwritten text using techniques like segmentation or bounding boxes to isolate relevant information.

Handwriting Recognition: Handwriting recognition is a key functionality, where the system should employ a trained CNN model to accurately classify and interpret handwritten characters or word from prescription images, accounting for variations in handwriting styles and pen types. Complementing this, the system must integrate OCR technology, such as Tesseract, to extract textual data from both handwritten and printed sections of the prescription, ensuring comprehensive text capture.

Optical Character Recognition (OCR): Another critical requirement is parsing the extracted information to identify and categorize essential elements such as patient names, medication details, dosages, and instructions, presenting them in a structured format. For cases where the system cannot accurately read the handwriting or when image quality is low, it should provide feedback, prompting users to re-upload or adjust the input.

User Interface: A user-friendly interface is essential, allowing healthcare professionals to upload prescriptions and view recognized text conveniently, with the option to manually edit extracted data in case of errors. To integrate seamlessly into existing workflows, the system must support exporting data in structured formats like JSON or CSV, enabling further processing or integration with healthcare management systems.

3.2.4 Non-Functional Requirement

Non-functional requirements (NFRs) define the qualities or constraints of a system, focusing on how the system performs rather than what it does. These requirements address aspects like performance, scalability, usability, reliability, and security, ensuring the system functions efficiently and meets user expectations.

Performance: The system should process and recognize prescriptions within 2-3 seconds per image, ensuring quick results for healthcare workflows.

Accuracy: The recognition system must achieve at least 95% accuracy in interpreting handwritten text, including medical terminologies, to minimize errors.

Scalability: The system should handle a growing number of users and increased data volume without significant performance degradation, supporting integration with hospital information systems.

Usability: The interface should be intuitive, requiring minimal training for healthcare staff. Features like drag-and-drop upload and real-time feedback enhance user experience.

Reliability: The system should function consistently with an uptime of at least 99.9%, ensuring it is available for critical healthcare operations.

Maintainability: The system should support modular updates, allowing for easy integration of newer CNN models or OCR technologies without disrupting current operations.

CHAPTER 4

SYSTEM DESIGN

4.1 Architectural Diagram

An architecture description is a formal description of a system, organized in a way that supports reasoning about the structural properties of the system. It defines the system componentsorbuildingblocksandprovidesaplanfromwhichproductscanbeprocured,and system developed, that will work together to implement the overall system.

Architectural diagram helps in:

- They help with comprehension: Architectural diagrams help convey complex information in a single image, like the saying, "A picture is worth a thousand words."

- They improve communication and collaboration: when we are working on anything that involves multiple people, there is always a risk of miscommunication between them .so it is important to standardize information, which is where the architectural diagrams are very useful.

On the backend, the Flask application acts as the processing hub. It employs Flask-CORS to handle cross-origin requests, enabling secure communication between the frontend and backend. Uploaded images are processed using Pillow for validation and pre-processing, followed by Tesseract OCR, which extracts text from the prescription. The extracted text is compared with data stored in a CSV file, managed using Pandas, to find matching medicines. This CSV file includes details such as medicine names, composition, uses, manufacturer, and optional image URLs. Temporary storage is used to handle uploaded files before processing, ensuring efficient handling of image data. The communication flow integrates all components: users upload files, backend processes the images and matches the data, and results are sent back to the frontend for display. The architecture ensures scalability, efficiency, and a user-friendly experience for recognizing and matching handwritten prescriptions.

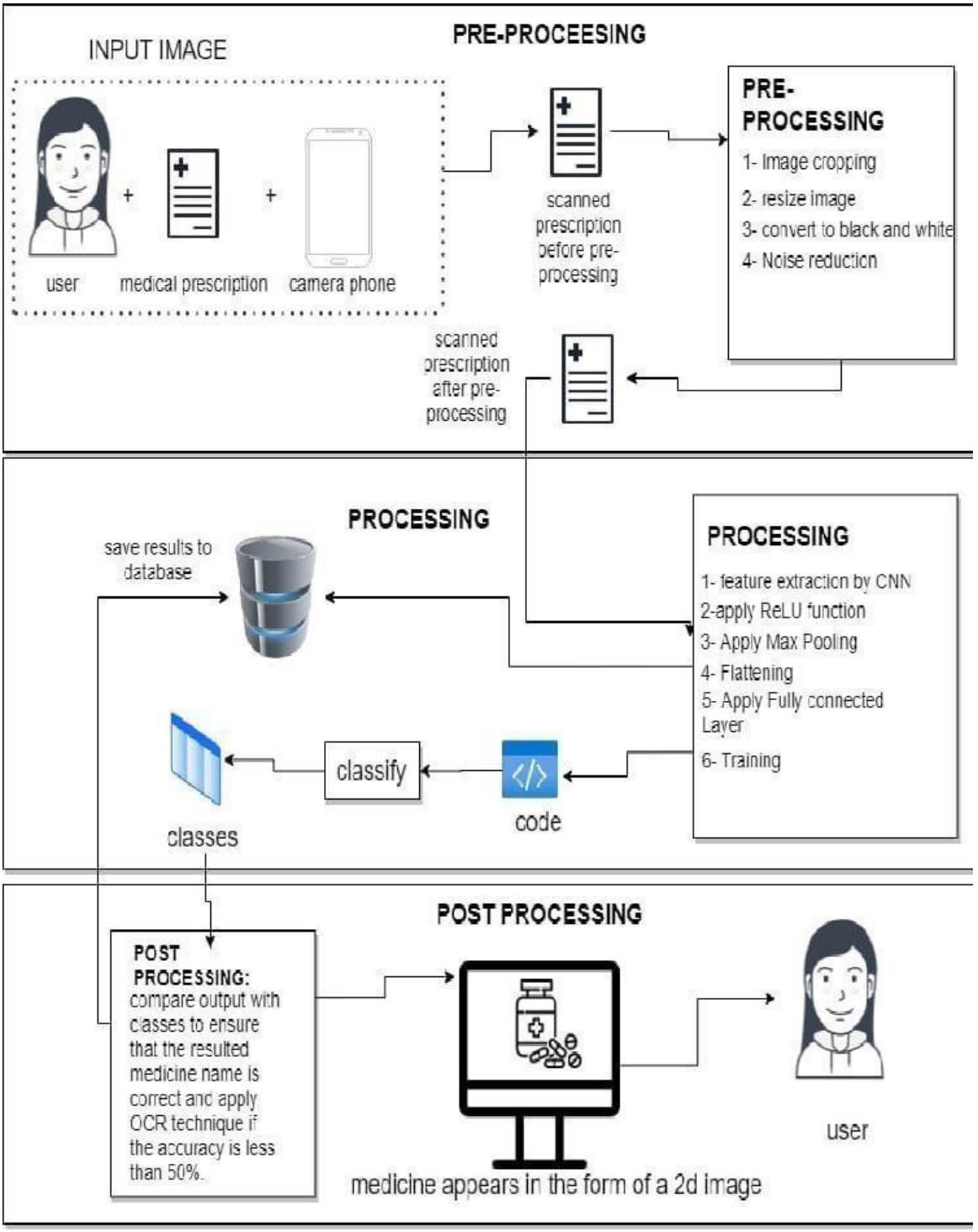

Figure4.1: Architectural Diagram of Proposed System

The prescription recognition application employs a modular and efficient architecture to integrate frontend, backend, and data processing. The React.js frontend provides an intuitive interface for users to upload prescription images. Through Axios, the frontend sends HTTP POST requests to the Flask backend for processing and dynamically displays

the results. A welcome screen introduces users to the application, while the main file upload interface manages interactions seamlessly..

4.2 Activity Diagram

The activity diagram for the prescription recognition app consists of a client-server model with clear separation between the frontend and backend components. The Frontend (React.js) handles the user interface, allowing users to upload prescription images. It communicates with the Backend (Flask API) via HTTP requests. The frontend uses the FileUpload component to capture the uploaded image, send it to the backend, and receive the processed response. The Backend consists of the FlaskApp, which performs the core processing. The uploaded image is passed through Pillow for validation and manipulation, then processed by Tesseract OCR for text extraction. The extracted text is compared with the medicine details stored in a CSV file, which is read and processed using Pandas. The backend then returns the matching results to the frontend, where they are displayed in the FileUpload component. Flask-CORS ensures smooth cross-origin communication between the React frontend and Flask backend. This architecture provides a clear, modular structure that separates concerns and facilitates easy maintenance and scalability.

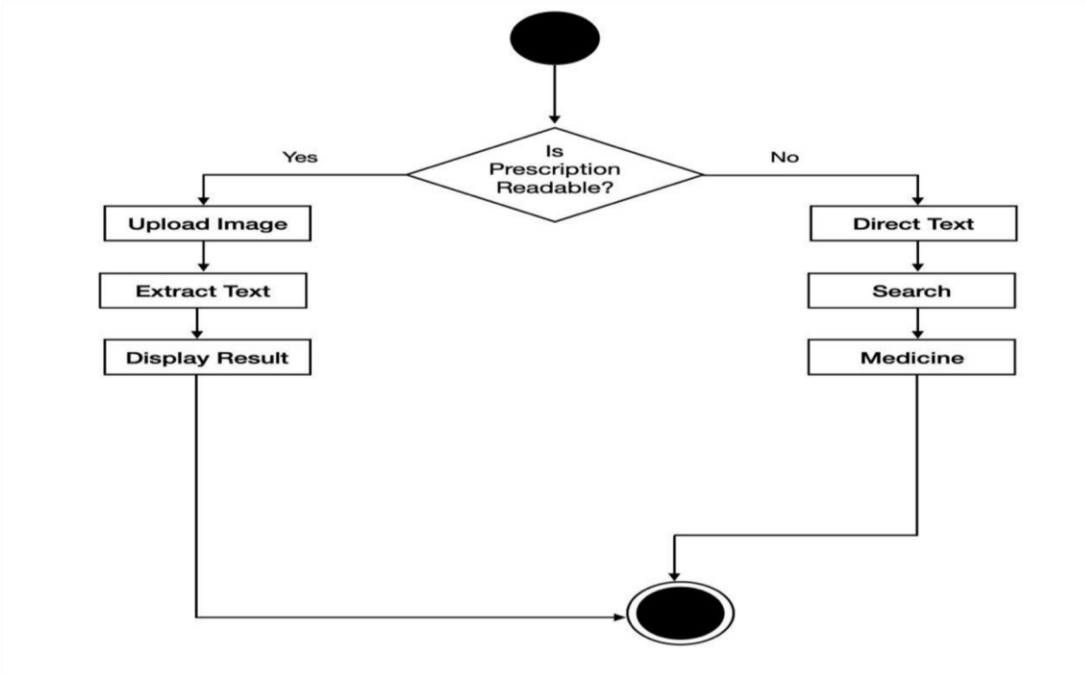

Figure4.2: Activity Diagram of Proposed System

4.3 Data Flow Diagram

The flow diagram for the prescription recognition app outlines the sequence of steps from the user's interaction with the frontend to the backend processing and response. Initially, the user uploads an image through the FileUpload component in the React app. This triggers an HTTP POST request to the FlaskApp backend, where the image is processed using Pillow to ensure it is a valid image file. The file is then passed to Tesseract OCR for text extraction. The extracted text is compared with the medicine names in the Pandas DataFrame loaded from a CSV file. If matches are found, they are compiled into a response, which is sent back to the frontend. The FileUpload component processes this response, updating the UI to display matched medicine details or an error message if no matches are found. The App component manages the overall navigation flow, toggling between the welcome screen and the main page based on user interaction

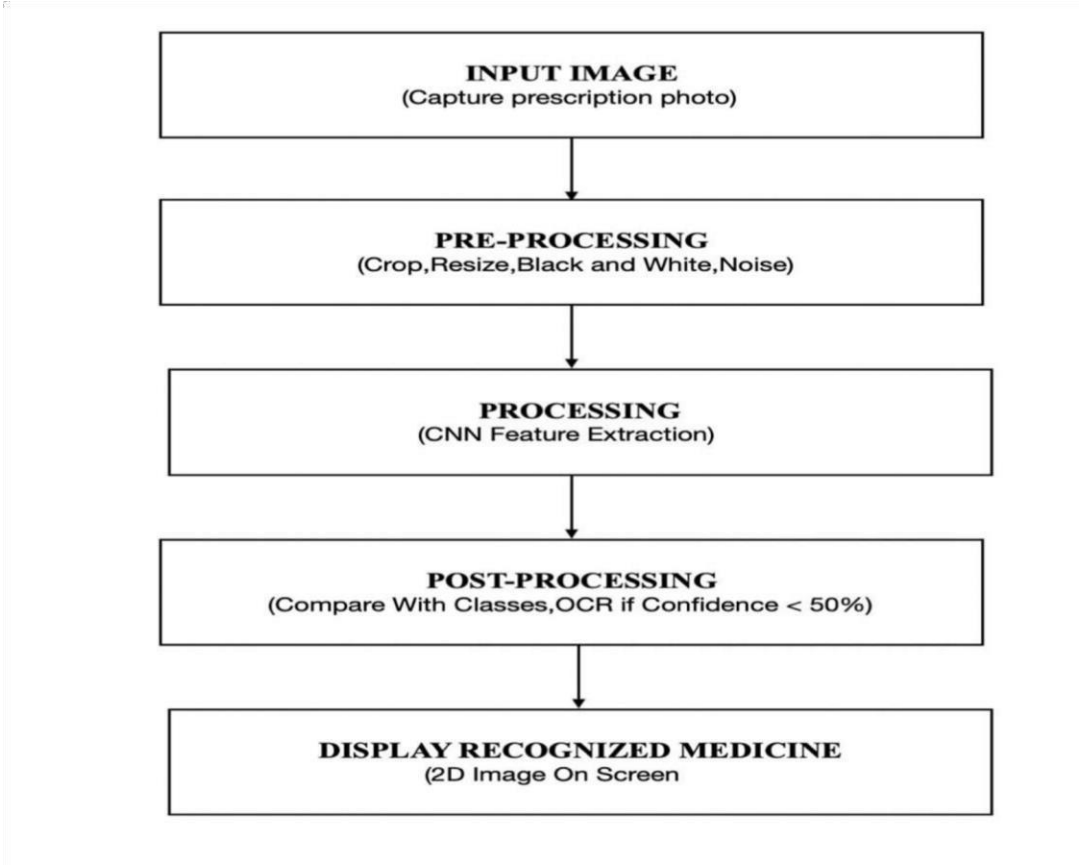

Figure 4.3: Data Flow Diagram of Proposed System

4.4 Sequence Diagram

The sequence diagram for the prescription recognition app illustrates the flow of interactions between the User, Frontend (React.js), Backend (Flask API), and external services like Tesseract OCR and Pandas. The process begins when the User selects an image file on the frontend. The Frontend (FileUpload) component captures the file and sends an HTTP POST request with the image to the Backend (FlaskApp). The Backend receives the image, and Pillow is used to open and validate it. The image is then passed to Tesseract OCR for text extraction. The extracted text is compared with the medicine names in the Pandas DataFrame, which is loaded from a CSV file containing medicine details. If matches are found, the Backend prepares a response with the matching results. This response is sent back to the Frontend, which processes the data and displays the matched medicines or error messages. The frontend may also handle user navigation between the welcome and upload screens.

4.5 Use Case Diagram

The use case diagram for the prescription recognition app highlights the interactions between the User, Frontend (React.js), and Backend (Flask API). The primary actor is the User, who interacts with the system to upload prescription images. The main use case for the user is Upload Prescription Image, where the user selects an image file on the frontend. The Frontend (FileUpload) component handles the file upload process by sending the image to the Backend (FlaskApp). The backend processes the image using Pillow and Tesseract OCR for text extraction. The extracted text is compared with the medicine data stored in a CSV file, using Pandas for data matching. If a match is found, the backend returns the matched medicines, which are displayed in the frontend as the Display Matched Medicines use case. If no matches are found, the system provides an error message. The User can also navigate between different screens (welcome and upload). This diagram represents the core functionalities and user interactions within the app.

CHAPTER 5

IMPLEMENTATION

5.1 Module Implementation

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, algorithm, or policy. Implementation refers to conversion of a new system design to an operation. An implementation plan is made before starting the actual implementation of the system. The new system may be totally new, replacing an existing.

manual or automated system or it may be a major modification to an existing system

5.1.1 Hardware Implementation

The hardware implementation of the prescription recognition app involves both client-side and server-side components. On the client side, the User's Device (such as a computer or smartphone) is responsible for uploading prescription images through a web browser. The hardware specifications of the user's device should be capable of handling basic web interactions, including file uploads, and running a modern web browser. The client-side React.js application does not require heavy processing power, but the device should have an active internet connection to communicate with the backend server.On the server side, the Backend (Flask API) runs on a machine that should meet the requirements for image processing and running Python-based libraries. The server must have enough memory and CPU power to handle image files efficiently, process them with Pillow, and perform Optical Character Recognition (OCR) using Tesseract OCR. For optimal performance, the server should be equipped with sufficient RAM and processing capacity to handle multiple simultaneous requests. Additionally, a database or file storage system (such as local storage or cloud storage) is necessary to store the medicine details CSV file used by Pandas for matching extracted text. If required, an internet connection for external APIs or database access is essential. This hardware setup ensures smooth operation of the app, from file upload to text extraction and data matching.

5.1.2 Software Implementation

The software implementation of the prescription recognition app involves a combination of frontend and backend technologies, working together to provide an efficient user

experience. On the frontend, the app is built using React.js, which allows the creation of dynamic and responsive components. The user interface includes a file upload component (FileUpload) where users can select prescription images. React's state management, through the useState hook, is used to track file selection, loading status, and the server response. The frontend communicates with the backend using axios to send HTTP POST requests for image processing and receive the response. It is also responsible for rendering the matched medicines or error messages returned from the backend.

On the backend, the app uses Flask, a lightweight Python web framework, to handle incoming requests and process the uploaded image. Pillow is utilized to open and manipulate image files, ensuring compatibility for OCR processing. Tesseract OCR, an open-source tool, is used to extract text from the images. The extracted text is then compared to a CSV file containing medicine details, which is loaded and processed using Pandas. Finally, Flask-CORS is implemented to allow cross-origin communication between the React frontend and Flask backend. This setup ensures smooth and efficient processing of prescription images and the display of matching medicine details.

5.2 Code Snippet

5.2.1 Code snippet for Flask backend

```
from flask import Flask, request, jsonify # type: ignore
from flask_cors import CORS # type: ignore
import pytesseract
from PIL import Image
import pandas as pd
import os
import numpy as np
import cv2
from tensorflow.keras.applications import VGG16 # type: ignore
from tensorflow.keras.applications.vgg16 import preprocess_input # type: ignore

app = Flask(_name_)
CORS(app)

# Configure Tesseract executable path
pytesseract.pytesseract.tesseract_cmd = r'F:\Program Files\Tesseract-OCR\tesseract.exe'

# Load the medicine details CSV
csv_path = "medicine_details.csv"  # Ensure this file is in the same directory
medicine_data = pd.read_csv(csv_path)
```

```
# Load the pre-trained VGG16 model
vgg16_model = VGG16(weights='imagenet', include_top=False)

def preprocess_with_cnn(image):
    """
    Preprocess the image using the pre-trained VGG16 CNN model.
    """
    # Resize image to match CNN input size (224x224 for VGG16)
    image = cv2.resize(image, (224, 224))

    # Convert to a 3-channel format if it's grayscale
    if len(image.shape) == 2:
        image = cv2.cvtColor(image, cv2.COLOR_GRAY2RGB)
    elif image.shape[2] == 1:
        image = cv2.merge([image, image, image])

    # Prepare the image for the CNN
    image = np.expand_dims(image, axis=0)
    image = preprocess_input(image)

    # Extract features using VGG16
    features = vgg16_model.predict(image)

    # Return the processed features (can be used as is or converted back to an image)
    return features

@app.route('/upload', methods=['POST'])
def upload():
    if 'file' not in request.files:
        return jsonify({"error": "No file provided"}), 400

    file = request.files['file']

    if file.filename == '':
        return jsonify({"error": "No file selected"}), 400

    try:
        # Save uploaded file temporarily
        file_path = os.path.join('temp', file.filename)
        os.makedirs('temp', exist_ok=True)
        file.save(file_path)

        # Load image and preprocess with CNN
        image = cv2.imread(file_path)
```

```
        cnn_features = preprocess_with_cnn(image)

        # Convert CNN features back to an image if necessary (for this example, we skip
it)
        # Alternatively, you can directly pass image or cnn_features to Tesseract

        # Perform OCR using Tesseract
        pil_image = Image.open(file_path)  # Using original image for OCR
        extracted_text = pytesseract.image_to_string(pil_image)
        os.remove(file_path)  # Remove the temp file

        # Match text to medicines
        matches = []
        input_prefix = extracted_text[:5].strip().lower()
        for _, row in medicine_data.iterrows():
            if row['Medicine Name'][:5].strip().lower() == input_prefix:
                matches.append(row.to_dict())

        response = {
            "matches": matches,
            "center_align": len(matches) == 1  # True if only one match
        }

        return jsonify(response), 200

    except Exception as e:
        return jsonify({"error": str(e)}), 500

if _name_ == '_main_':
    app.run(debug=True)
```

CHAPTER 6

RESULTS AND DISCUSSION

6.1 Result

The results of the prescription recognition app are displayed based on the image uploaded by the user. After the user selects and uploads a prescription image, the backend processes the image using Optical Character Recognition (OCR) with Tesseract to extract text. The extracted text is compared with medicine names stored in a CSV file. If the first five characters of the extracted text match any medicine name in the database, the system returns a list of matching medicines, including details such as the medicine name, composition, uses, and manufacturer. These results are displayed on the frontend in a well-organized format, showing the matched medicines in a card-style layout. If no matches are found, an error message is displayed, indicating that no medicines were identified. This process ensures a quick, efficient way for users to identify prescribed medicines from images, enhancing the usability of the app for medical professionals or patients.

6.2 Snapshots

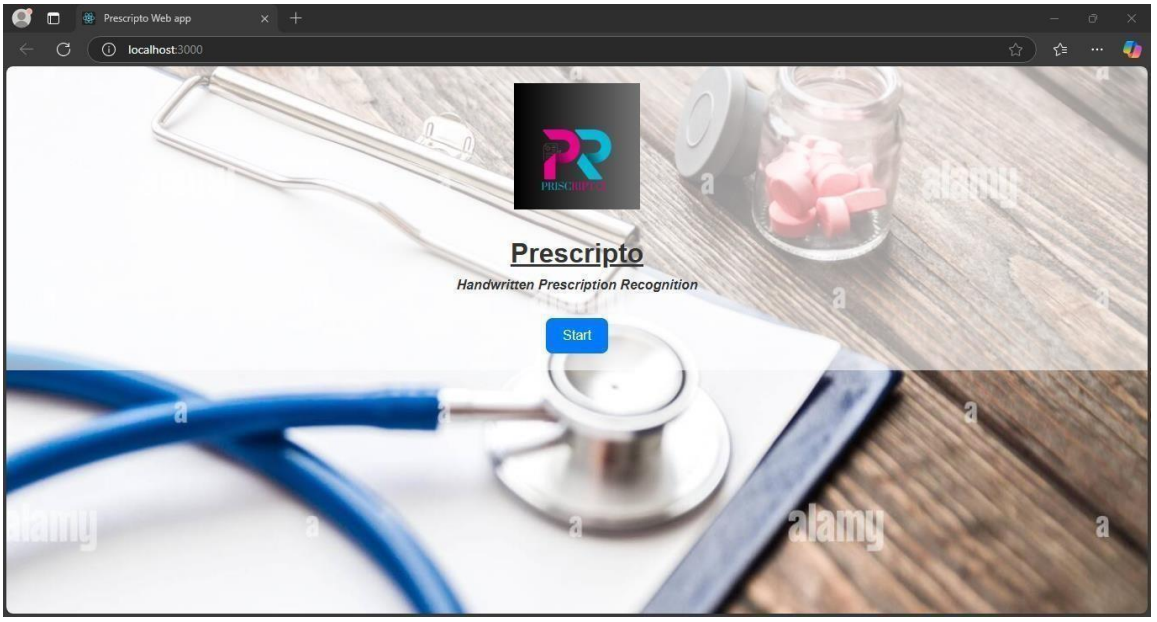

Fig 6.2.1: Home Page Interface

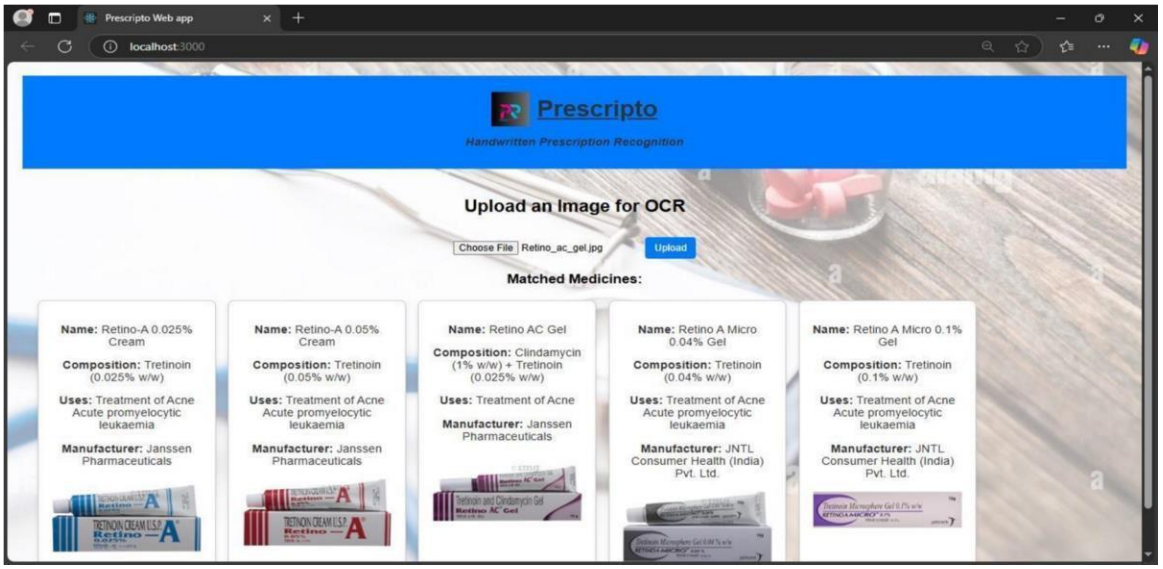

Fig 6.2.2: Matched Medicines Displayed After Image Upload

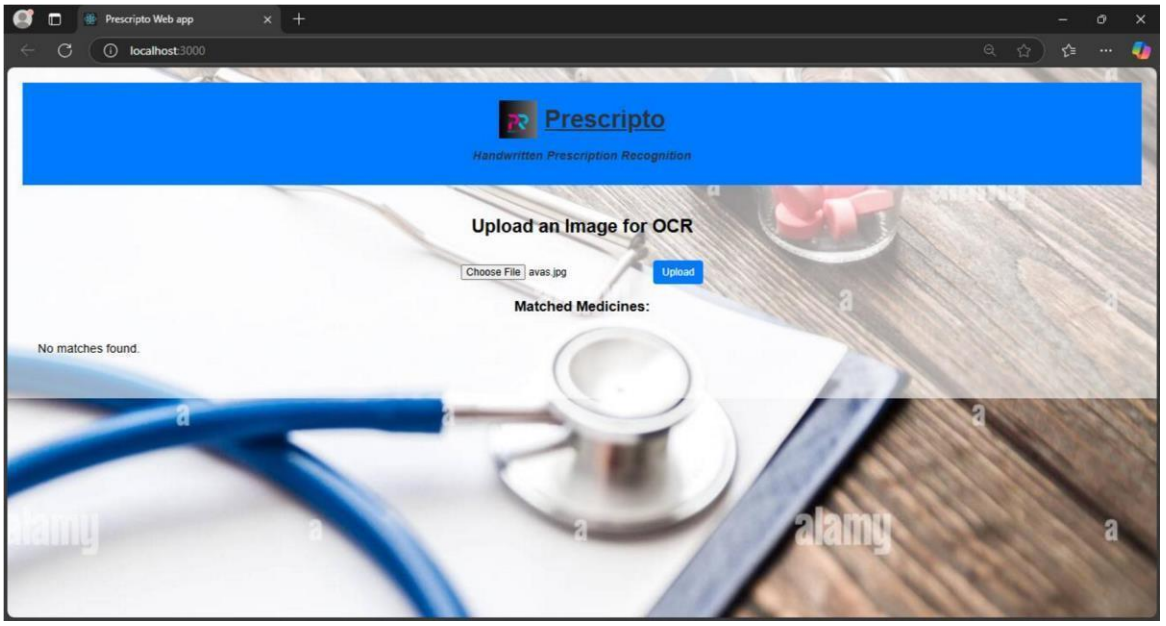

Fig 6.2.3: No Matches Found After Image Upload

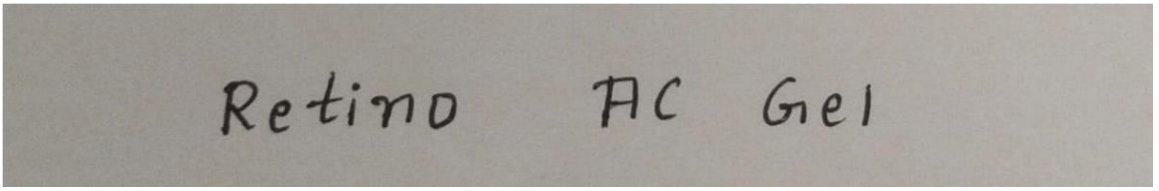

Fig 6.2.4: Test Input Image for OCR System

CHAPTER 7

CONCLUSION

7.1 Conclusion

In conclusion, the prescription recognition app successfully integrates modern technologies to offer an efficient solution for identifying medicines from handwritten prescriptions. By combining React.js for the frontend with Flask and Pandas for the backend, the app provides a seamless user experience, allowing users to upload prescription images and retrieve matching medicine details. The use of Tesseract OCR ensures accurate text extraction from images, while Pillow handles image processing to ensure compatibility with the OCR tool. Pandas efficiently manages the medicine database for matching extracted text. The implementation of Flask-CORS facilitates smooth communication between the frontend and backend, even when they run on different domains. This app streamlines the process of prescription identification, offering a reliable, scalable solution for users, such as healthcare professionals and patients. By automating text extraction and matching, it saves time and enhances the accuracy of identifying prescribed medicines.

7.2 Future Work

For future work, the prescription recognition app can be further enhanced in several ways. First, improving the OCR accuracy could be a priority by integrating more advanced OCR models or deep learning techniques to handle various handwriting styles and fonts more reliably. The app could also be expanded to support multiple languages and include advanced image pre-processing techniques to reduce errors caused by poor image quality or distortion. Additionally, integrating a cloud-based database for medicine information would allow for easier updates and scalability, enabling the app to access the latest medicine data in real-time. To improve the app's usability, implementing features like user authentication could allow users to save their search history or manage medicine lists. Another potential area for development is mobile app support, making the solution more accessible to a broader user base. Finally, adding voice recognition for prescriptions could further streamline the identification process in busy healthcare settings.

CHAPTER 8

REFERENCES

1. Esraa Hassan, Habiba Tarek, Mai Hazem, Shaza Bahnacy, Lobna Shaheen, Walaa H. Elashmwai. "Handwritten Recognition, particularly in the context of doctors' handwriting." Faculty of Computer Science, Misr International University, Cairo, Egypt.
2. Sandhya P., Rama Prabha K.P., Jayanthi R., V. Sujatha, Asha N., M. B. Benjulaanbu Malar. "Handwritten Medicine Name Recognition Using CNN and Hybrid Algorithms." VIT, Chennai, Tamil Nadu, India.
3. Swasthika, Rithika C. Jain, Sharanya, Shubhashree, Annappa Swamy D. R. "Exploring Hybrid Approaches for Handwritten Recognition." Mangalore Institute of Technology and Engineering, Moodabidri, Karnataka, India.
4. Maheen Islam, Mahamudul Hasan, Nishat Vasker. "Enhancing Handwritten Recognition Systems with Machine Learning Techniques." Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh.
5. L. Au et al., "Recognition of Handwritten Chinese Characters Based on Concept Learning," *IEEE Explore*, 2019
6. Y. Zhu et al., "SCUT-EPT: New Dataset and Benchmark for Offline Chinese Text Recognition in Examination Paper," *IEEE Explore*, 2019
7. M. Rajalakshmi, P. Saranya, and P. Shanmugavadivu, "Pattern Recognition - Recognition of Handwritten Document Using Convolutional Neural Networks," *IEEE Explore*, 2019.
8. M. Eltay, A. Zidouri, and I. Ahmad, "Exploring Deep Learning Approaches to Recognize Handwritten Arabic Texts," *IEEE Explore*, 2020.
9. T. M. Ghanim, M. I. Khalil, and H. M. Abbas, "Comparative Study on Deep Convolution Neural Networks (DCNN)-Based Offline Arabic Handwriting Recognition," *IEEE Explore*, 2020.
10. T. Sahlol et al., "Handwritten Arabic Optical Character Recognition Approach Based on Hybrid Whale Optimization Algorithm with Neighborhood Rough Set," *IEEE Explore*, 2020.