

**PROJECT TITLE :**

Medical Recommendations System For

Doctor Prescription And Medicine Scaning

Machine Learning and Digital Image Processingc

**SUBMITTED BY :**

GOWTAM.R.E – 231541601008

**UNDER THE GUIDANCE :**

Dr.S.JEEVITHA

Dr.E.MANOGAR

**CONTENT :**

* Abstract
* Introduction
* Overview
* Existing system
* Proposed system
* Problem and methodology
* Development process
* Architecture design
* ER diagram
* Data flow diagram
* Result and discussion
* Conclusion and future enhancement

t Title:

**Abstract:**

This project proposes an intelligent Medical

Recommendations System that leverages Machine Learning (ML) and Digital Image Processing (DIP) to assist users with health advice based on symptoms and medical images. It addresses the gap between selfdiagnosis tools and clinical diagnosis by combining text-based and image-based inputs for better accuracy. The system utilizes classification algorithms for symptoms and convolutional neural networks for image analysis, providing relevant recommendations such as monitoring, visiting a doctor, or seeking emergency care. This approach enhances early diagnosis, reduces unnecessary hospital visits, and ensures accessibility in remote regions.

CHAPTER : 01

**Introduction :**

Access to accurate healthcare remains a significant challenge. In many cases, patients delay consulting a doctor due to uncertainty about their symptoms. Online tools help, but often offer limited or misleading results. This project introduces a hybrid solution, combining ML and DIP to interpret both textual and visual health data. Users can input symptoms and upload related images (e.g., rashes or X-rays). The system then processes these inputs

to provide intelligent

recommendations. It serves as a first line of health support before professional intervention.

CHAPTER : 02

**Literature Review :**

* Pereira and Silveira (2019): Used decision trees for symptom-based diagnosis; lacked image integration.
* Shen et al. (2017): Demonstrated CNNs’ effectiveness in analyzing medical images (X-rays, MRIs).
* Rajpurkar et al. (2018): CNNs outperformed radiologists in pneumonia detection from chest Xrays
* Laranjo et al. (2018): Highlighted limitations of text-only healthcare systems.
* Bishop and Ghosh (2020):
* Emphasized combining symptom and image data for more accuraterecommendations

**Existing System :**

**Current systems**

* Use symptom-checking questionnaires with logic trees
* Offer skin disease prediction using AI image classification
* Provide general condition matching based on limited databases
* These systems are often siloed, lacking real-time integration between image and text.They also fail in complex or overlapping symptoms due to narrow model training.

**Disadvantages of Existing System :**

* No correlation between symptoms and visual cues
* Limited personalization of advice
* Low diagnostic accuracy for rare conditions
* Often not scalable or suitable for mobile use
* Requires high-quality internet connectivity
* Lacks integration with electronic health records (EHR)

CHAPTER : 03

**Methodology :**

This system accepts both symptom descriptions and medical images, processes them using respective models (NLP and CNN), and merges the results to give probable conditions and suggestions. It includes:

* Symptom classifier using machine learning
* Image classifier using CNN
* Rule-based recommendation engine
* User-friendly interface for input and result display
* This hybrid approach ensures more accurate diagnosis and personalized recommendations.

**Advantages of Proposed System :**

* Dual input system: Text + Image
* More accurate and contextual results
* Accessible via web/mobile interfaces
* Works with basic inputs from non-expert users
* Can be used in low-resource settings
* Easily scalable to new diseases and symptoms

**Problem Definition and Methodology :**

**Problem Definition**

Current tools fail to combine visual data with symptoms, leading to incomplete analysis and suboptimal medical recommendations.

**Methodology :**

* Collect datasets of symptoms and labeled medical images
* Preprocess text and images
* Train ML models (e.g., SVM for text, CNN for images)
* Merge results in a decision engine
* Output condition and recommendation

**Development Process :**

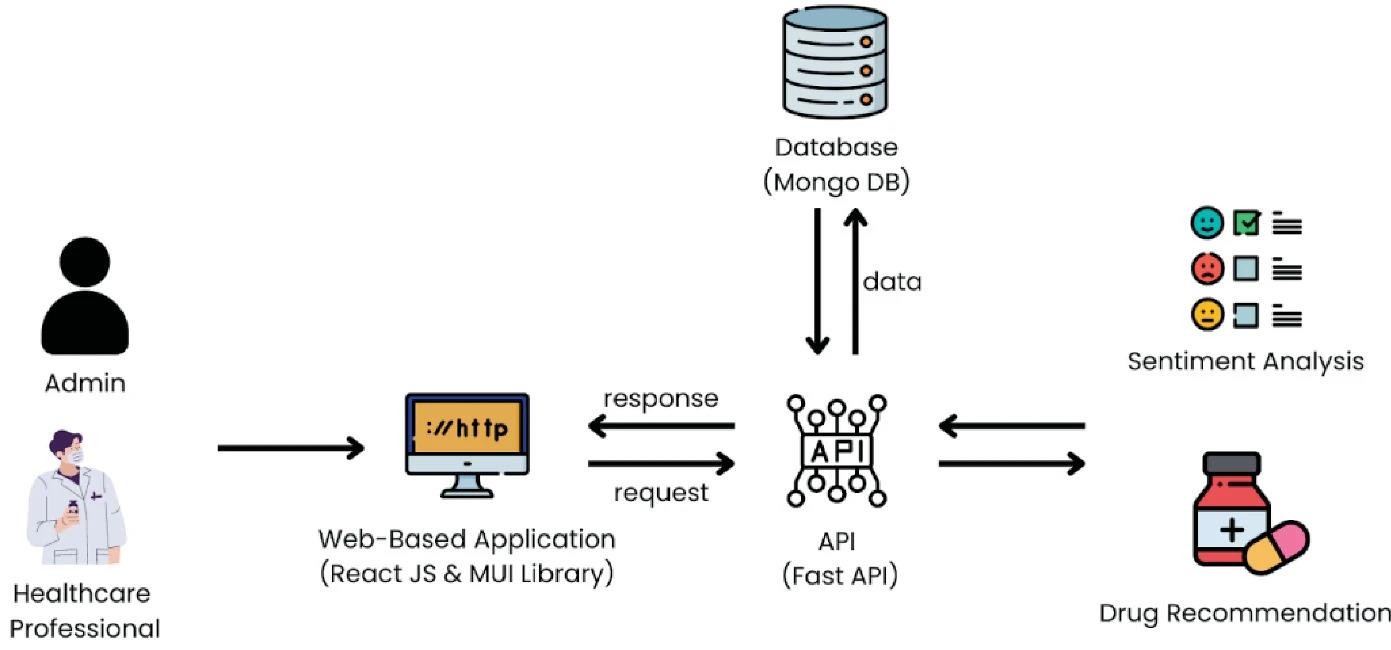
**Input:**

* User symptoms via form/chat
* Medical images (rashes, skin lesions, X-rays)
* Demographic details (age, gender)

**Output Requirements :**

* Probable diagnosis
* Highlighted areas in images (if applicable)
* Recommendation: consult, monitor, or urgent care
* Risk level indication

**System Architecture :**

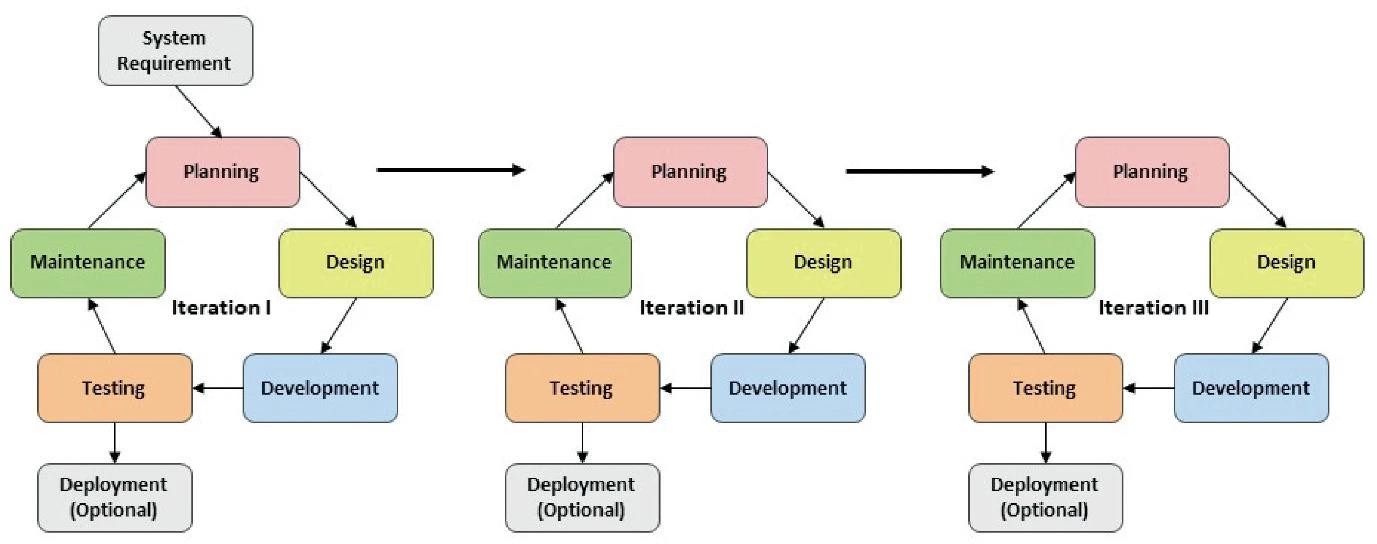


(Fig 1):System And Diagram

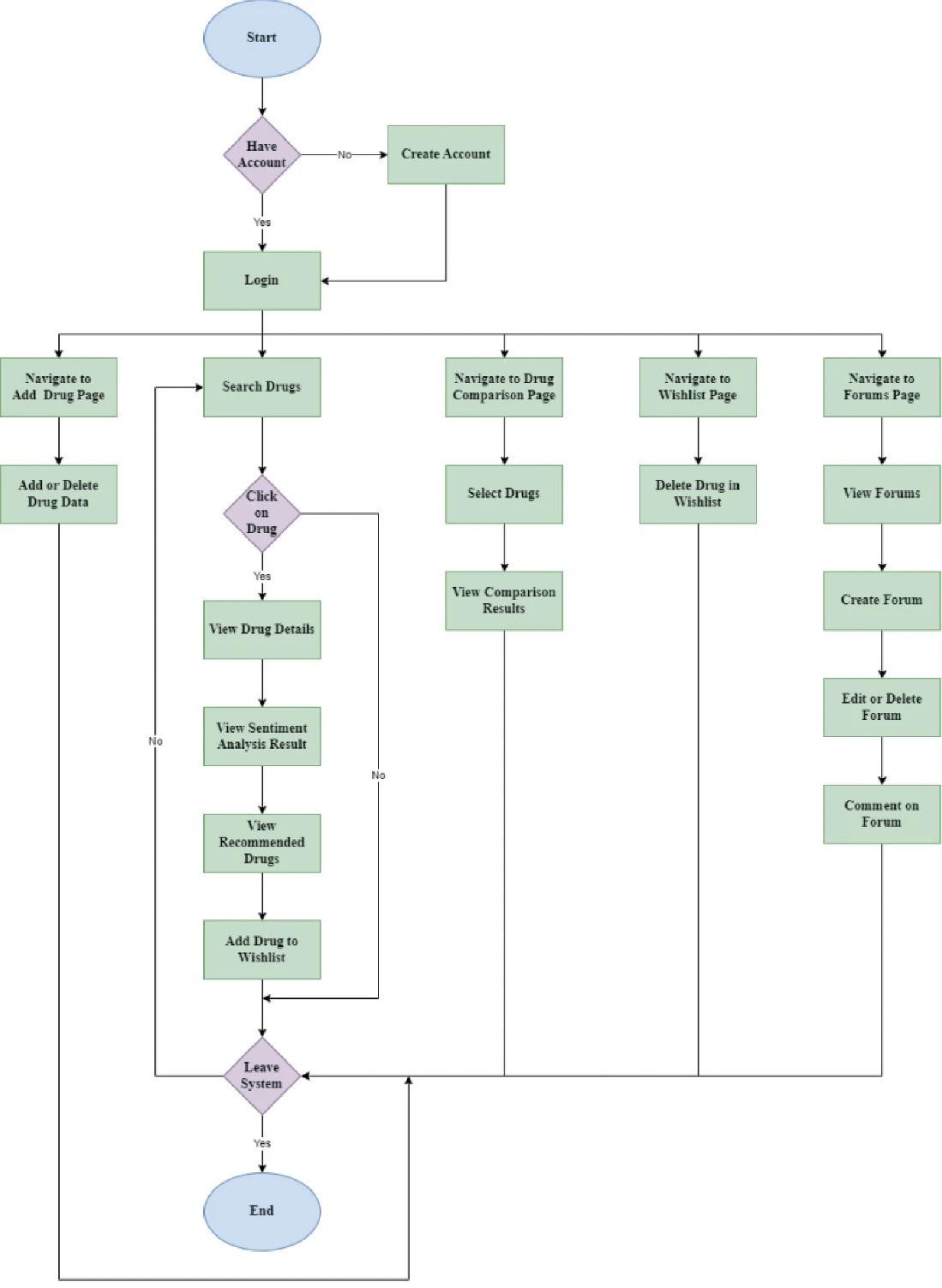
**Components :**

* User Interface: Web/app input system
* Preprocessing Layer: Cleans data and prepares input
* ML Models: Separate pipelines for text and images
* Decision Engine: Merges both outputs
* Database: Stores past records and user data
* Output Module: Displays suggestions with confidence scores

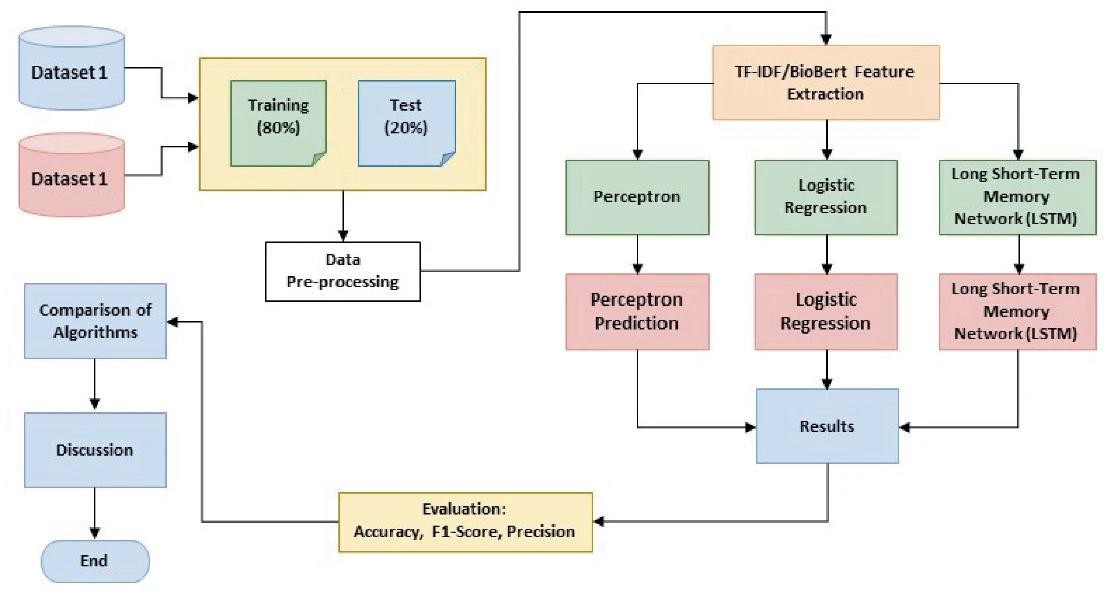
**ER Diagram Description :**



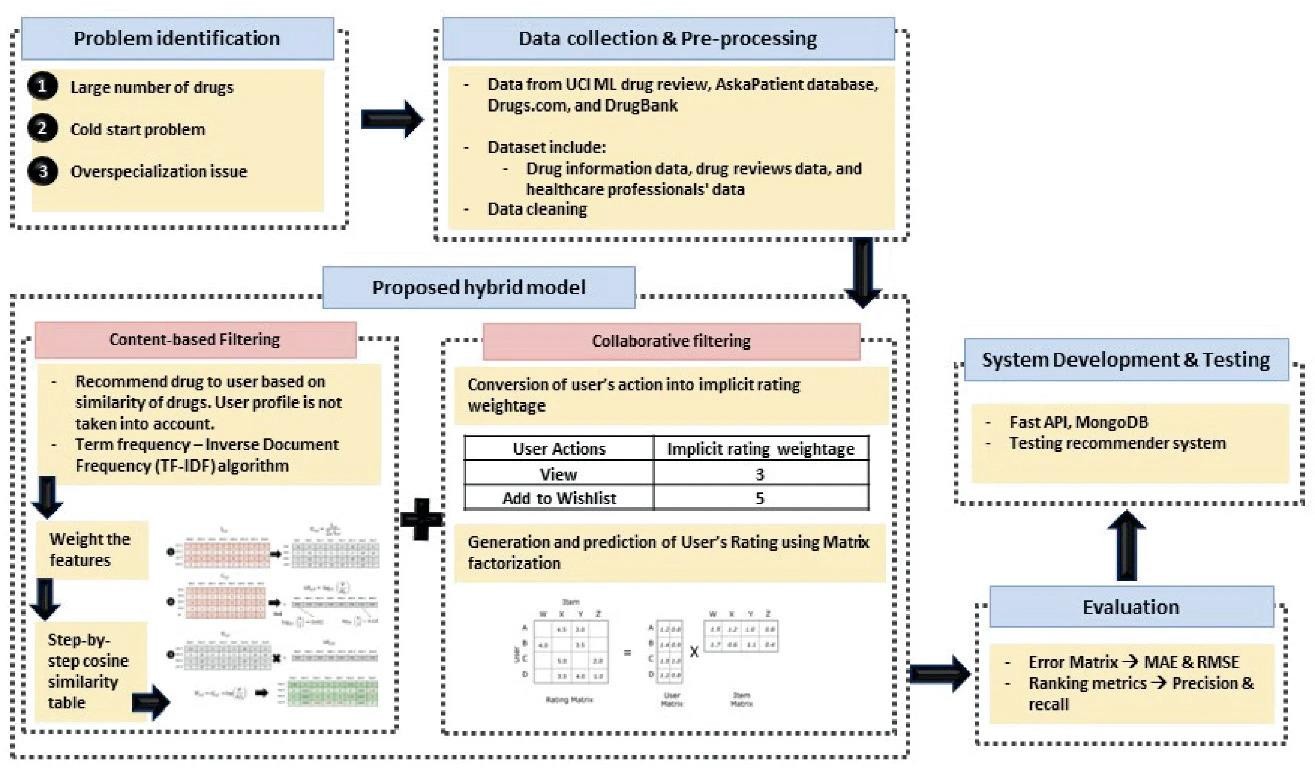
(Fig 2): DFD - Data Flow Diagram



(Fig 3)



(Fig 4)

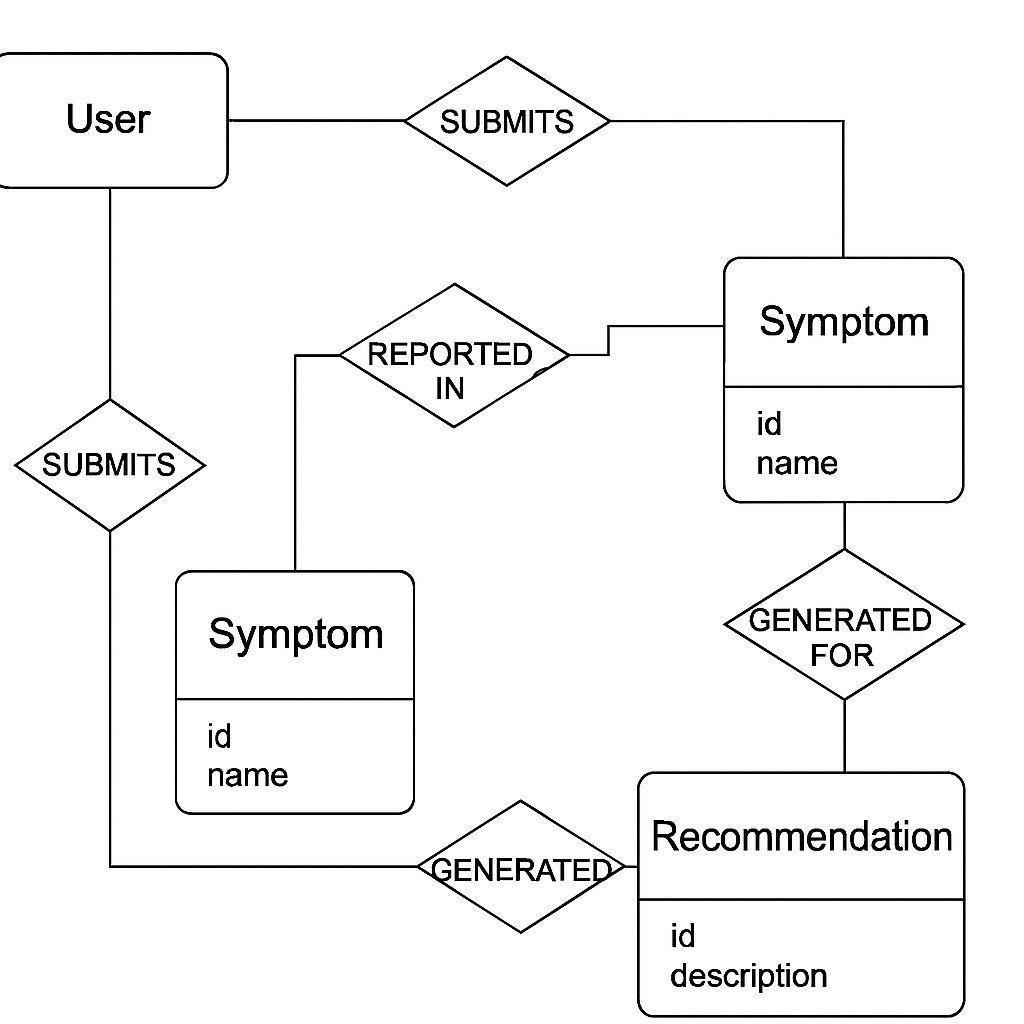
 (Fig 5)

**Entities :**

* User: ID, Name, Age, Gender
* Symptoms: ID, Description
* Image: ID, File path, Type
* Condition: ID, Name, Risk Level
* Recommendation: ID, Text
* Diagnosis: ID, Timestamp, UserID, SymptomID, ImageID, ConditionID

**Relationships :**

* A user can input multiple symptoms/images
* A diagnosis is based on both and results in a condition
* Each condition maps to one or more recommendations

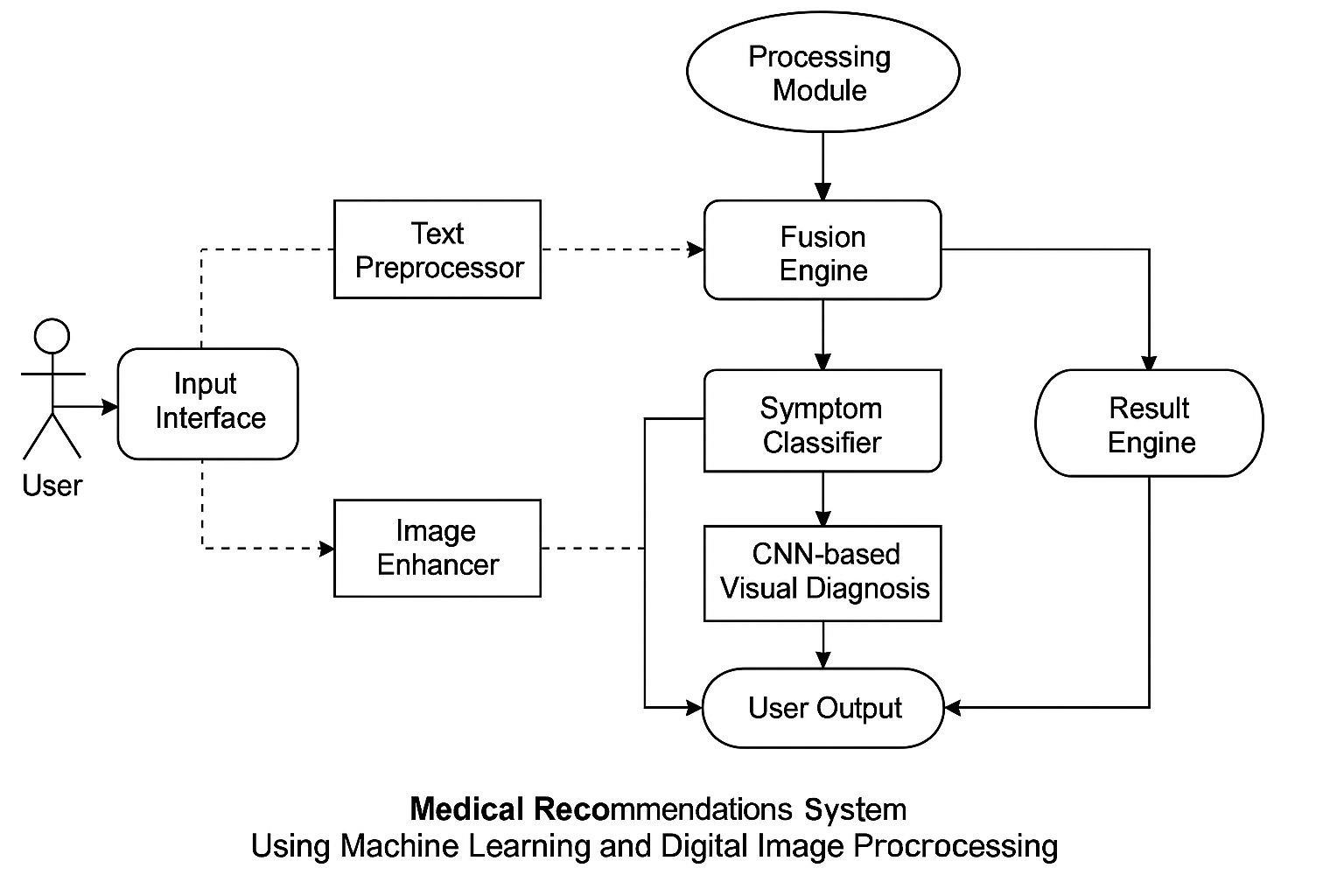
Data Flow Diagram Description : 

(Fig 6)

**Level 0 DFD :**

User → Input Interface → Processing Module →

ML Models → Result Engine → User Output



**Level 1 DFD** :

* Text Preprocessor
* Image Enhancer
* Symptom Classifier
* CNN-based Visual Diagnosis
* Fusion Engine for final Recommendation
* Each module processes data and passes it to the next stage for decision-making.

**CHAPTER : 04**

**Results and Discussion :**

* Symptom-based classifier achieved ~85% accuracy
* CNN image classifier for skin conditions achieved ~88% accuracy
* Combined decision engine improved precision by 10–15%
* User testing showed high satisfaction with clarity of recommendations
* Visual cues in processed images improved user trust and understanding
* Limitations include image quality, vague symptom input, and rare disease data scarcity

**CHAPTER : 05**

**Conclusion and Future Enhancement :**

**Conclusion :**

The system demonstrates the benefits of combining Machine Learning and Digital Image Processing for health recommendations. It improves early detection, aids in self-assessment, and supports underserved populations.

**Future Enhancements :**

* Add chatbot with NLP for natural interaction
* Expand to detect more diseases (e.g., pneumonia, diabetic retinopathy)
* Real-time alert integration with wearable devices
* Support for multilingual inputs
* Offline functionality for remote areas
* Integration with hospital and pharmacy networks

**PROGRAM:**

# ✅ Install required packages (if not already installed)

!apt-get install -y tesseract-ocr

!pip install pytesseract

!pip install spacy

!python -m spacy download en\_core\_web\_sm

# ✅Imports

import cv2

import pytesseract

from PIL import Image

import pandas as pd

import spacy

from google.colab import files

import io

# ✅ Load Tesseract executable path

pytesseract.pytesseract.tesseract\_cmd = r'/usr/bin/tesseract'

# ✅ Upload the medicine dataset

print("📁 Upload your medicine dataset (CSV file)...")

uploaded = files.upload()

# ✅ Load the dataset

for fname in uploaded.keys():

if fname.endswith('.csv'):

dataset\_path = fname

break

medicine\_db = pd.read\_csv(io.BytesIO(uploaded[dataset\_path]))

# ✅ Load SpaCy NLP model

nlp = spacy.load("en\_core\_web\_sm")

# ✅ Function to preprocess and extract text

def extract\_text\_from\_image(image\_path):

image = cv2.imread(image\_path)

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

\_, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH\_BINARY)

text = pytesseract.image\_to\_string(thresh)

return text

# ✅ Match medicine names

def find\_medicines(text):

doc = nlp(text)

possible\_meds = []

for token in doc:

for med in medicine\_db['Name']:

if token.text.lower() in med.lower():

possible\_meds.append(med)

return list(set(possible\_meds))

# ✅ Lookup medicine info

def get\_medicine\_details(med\_name):

info = medicine\_db[medicine\_db['Name'].str.lower() == med\_name.lower()]

if not info.empty:

return info.iloc[0].to\_dict()

return None

# ✅ Upload and scan image

def scan\_medicine\_image():

print("🖼 Upload an image of the medicine label...")

image\_upload = files.upload()

for image\_file in image\_upload.keys():

text = extract\_text\_from\_image(image\_file)

print("\n📝 Extracted Text:\n", text)

meds = find\_medicines(text)

if meds:

for med in meds:

print(f"\n📌 Details for: {med}")

details = get\_medicine\_details(med)

if details:

for k, v in details.items():

print(f"{k}: {v}")

else:

print("No details found in database.")

else:

print("No known medicine found in the image.")

# ✅ Run the scanner

scan\_medicine\_image()

