ABSTRACT

Psoriasis is a chronic autoimmune skin disease characterized by rapid skin cell proliferation, leading to scaling, inflammation, and discomfort. Early detection and effective monitoring are critical for managing its progression and improving patient outcomes. However, diagnosis and treatment planning often rely on subjective visual assessments by dermatologists, which can lead to inconsistencies. This project explores the application of Artificial Intelligence (AI) in the diagnosis, classification, and monitoring of psoriasis through image-based analysis.

Using deep learning techniques, particularly convolutional neural networks (CNNs), the project develops a model capable of accurately identifying psoriatic lesions from skin images and distinguishing them from other skin conditions. The AI system is trained on annotated dermatological image datasets and optimized for accuracy, sensitivity, and specificity. Additionally, the model can assess the severity of psoriasis, supporting clinicians in treatment decisions and follow-up.

This AI-driven approach aims to assist healthcare professionals in achieving faster, more consistent diagnoses, especially in regions with limited access to dermatological care. The integration of AI into dermatology holds the potential to revolutionize psoriasis management by enhancing diagnostic precision, reducing clinical workload, and ultimately improving patient care.

|  |  |
| --- | --- |
| **CONTENTS** |  |
| **CHAPTERS** | **PAGE NO** |
| Abstract | I |
| List of Figures | II |
| **1. INTRODUCTION** | 1-2 |
| 1.1 Problem Definition |  |
| 1.1.1 Background | 2 |
| 1.1.2 Problem Statement | 3 |
| 1.1.3 Project Significance | 3 |
| 1.2 Objective & Scope | 3 |
| **2. LITERATURE SURVEY** | 4-5 |
| **3. EXISTING SYSTEM** | 6 |
| 3.1 Disadvantages of Existing Systems | 7 |
| **4. PROPOSED SYSTEM** | 8 |
| 4.1 Advantages of Proposed System | 9 |
| **5. SYSTEM ARCHITECTURE** | 10-11 |
| 5.1 UML Diagram | 12-13 |
| 5.2 Use Case Diagram | 14 |
| 5.3 Sequence Diagram | 15 |
| 5.4 Class Diagram | 16 |
| 5.5 Activity Diagram | 17 |
| **6. SYSTEM SPECIFICATIONS** |  |
| 6.1 Hardware Requirements | 18 |
| 6.2 Software Requirements | 18 |
| **7. SYSTEM DESIGN** | 19 |

|  |  |
| --- | --- |
| 7.1 Logical Design | 19 |
| 7.2 Physical Design | 19-20 |
| 7.3 Algorithm Complexity | 20-25 |

1. **IMPLEMENTATION**
   1. [Implementation Steps 26-27](#_TOC_250014)
   2. [Sample Code 27-32](#_TOC_250013)
2. [SYSTEM TESTING](#_TOC_250012)
   1. [Introduction 33](#_TOC_250011)
   2. [Levels of Testing 33-36](#_TOC_250010)
3. [RESULTS AND SCREENSHOTS 37](#_TOC_250009)
4. [CONCLUSION 45](#_TOC_250002)
5. [FUTURE SCOPE 46](#_TOC_250001)
6. [REFERENCES 47-49](#_TOC_250000)

**LIST OF FIGURES**

|  |  |
| --- | --- |
| **TITLE** | **PAGE NO** |
| 1.1 Architecture | 10 |
| 1.2 UML Diagram | 12 |
| 1.3 Use Case Diagram | 14 |
| 1.4 Sequence Diagram | 15 |
| 1.5 Class Diagram | 16 |
| 1.6 Activity Diagram | 17 |
| 1.7 Decision Tree | 21 |
| 1.8 Random Forest Algorithm | 22 |
| 1.9 Naive Bayes Algorithm | 23 |
| 2.0 K-Nearest Neighbors (K-NN) Algorithm | 24 |
| 2.1 Convolutional Neural Network (CNN) | 25 |
| 2.2 Multi Disease predictor web app | 37 |
| 2.3 Breast Cancer Predictor | 38 |
| 2.4 Diabetes Predictor | 39 |
| 2.5 Heart Disease Predictor | 40 |
| 2.6 Kidney Disease Predictor | 41 |
| 2.7 Liver Disease Predictor | 42 |
| 2.8 Interface for Diabetes Predictor | 43 |
| 2.9 Snapshot of output after prediction | 43 |
| 3.0 Model Accuracies Comparison | 44 |

# INTRODUCTION

Psoriasis is a chronic, non-contagious autoimmune disease that primarily affects the skin, causing red, scaly patches that can be painful and debilitating. It affects millions of people worldwide and can significantly impact quality of life, especially when it becomes severe or is not properly managed. While the exact cause of psoriasis is still not fully understood, it is widely accepted that genetic and environmental factors contribute to its onset and progression. Early and accurate diagnosis is essential for effective management, as untreated or misdiagnosed psoriasis can lead to complications such as psoriatic arthritis and other systemic issues.

Traditionally, psoriasis diagnosis and monitoring rely heavily on visual examination by dermatologists, often supported by biopsy in unclear cases. However, this process can be subjective, time-consuming, and prone to variability between clinicians. In many regions, especially in rural or under-resourced areas, there is also a shortage of trained dermatologists, which delays diagnosis and treatment.

In recent years, **Artificial Intelligence (AI)** has emerged as a transformative technology in healthcare, offering tools for faster, more accurate, and consistent medical assessments. AI, particularly through **deep learning and computer vision**, has shown great promise in dermatology by analyzing medical images to detect and classify various skin conditions, including melanoma, eczema, and psoriasis. Convolutional Neural Networks (CNNs), a type of deep learning architecture, are especially effective in recognizing patterns in images and have been successfully applied in many medical imaging applications.

This project investigates the use of AI for the **automated detection and classification of psoriasis from skin images**. By training a model on a dataset of dermatological images, the goal is to develop a system that can not only identify psoriatic lesions but also evaluate their severity. Such a system can support dermatologists in making faster, more accurate diagnoses and can serve as a decision-support tool in clinical settings.

The use of AI in diagnosing psoriasis has the potential to:

* **Improve diagnostic accuracy**
* **Reduce the burden on healthcare systems**
* **Provide support in underserved regions**
* **Enable remote and teledermatology applications**
* **Facilitate early intervention and personalized treatment planning**

As the world continues to embrace digital health technologies, the integration of AI into dermatological practice could play a vital role in transforming the way psoriasis and other skin diseases are diagnosed and managed.

## Problem Definition

### Background

Psoriasis is an autoimmune skin disease characterized by an accelerated growth cycle of skin cells, resulting in thickened, inflamed, and scaly patches on the skin. It affects people of all ages and has no permanent cure, but with proper treatment, its symptoms can be managed effectively. Accurate and early diagnosis plays a crucial role in minimizing flare-ups, selecting appropriate therapies, and improving patient quality of life.

With the rapid advancement of AI and machine learning, especially in the field of **medical image analysis**, there is an opportunity to overcome these challenges. AI models can be trained on thousands of labeled skin images to learn features and patterns indicative of psoriasis. These systems can provide consistent, fast, and scalable diagnostic support.

This background lays the foundation for developing an AI-assisted diagnostic tool that can recognize psoriasis from skin images, classify its type and severity, and help bridge the healthcare gap, especially in areas with limited medical resources.

### Problem Statement

Despite being a common and chronic skin condition, psoriasis often goes misdiagnosed or undiagnosed due to the reliance on subjective visual assessments by dermatologists. This problem is further worsened in regions with limited access to specialized healthcare. Traditional methods of diagnosis are time-consuming, inconsistent, and not scalable. There is a critical need for an AI-based system that can accurately detect and classify psoriasis from skin images, assisting healthcare providers in delivering faster, more reliable, and accessible diagnosis and monitoring.

### Project Significance

This project holds significant value in the intersection of healthcare and artificial intelligence. By developing an AI-based system for detecting and classifying psoriasis from skin images, the project aims to:

* **Improve diagnostic accuracy** by reducing human error and subjectivity.
* **Enable early detection**, which is critical for effective treatment and management of psoriasis.
* **Support dermatologists** by acting as a decision-aid tool, especially in high-patient-load environments.
* **Expand access to care** in rural or underserved areas through remote and automated diagnosis.
* **Promote consistency** in assessing disease severity, helping monitor patient progress over time.

## Objective & Scope

**Objectives:**

* Develop an AI model to detect and classify psoriasis from skin images.
* Assist doctors in diagnosis and severity assessment.
* Improve accessibility for remote and underserved areas.

**Scope:**

* Focus on image-based psoriasis detection using deep learning.
* Analyze photos from smartphones or clinical cameras.
* Serve as a diagnostic aid, not a replacement for doctors.
* Use available datasets for training and testing.

# LITERATURE SURVEY

Early Psoriasis is a chronic autoimmune skin disorder that has been extensively studied in dermatology. Traditional diagnosis relies on clinical observation and histopathological examination, which can be time-consuming and subjective. In recent years, the integration of Artificial Intelligence (AI), especially **deep learning**, has revolutionized the approach to skin disease diagnosis.

**Deep Learning and Dermatology:**  
The success of AI in skin disease classification was notably demonstrated by Esteva et al. (2017), who trained CNNs on large datasets of skin images, achieving accuracy comparable to expert dermatologists. This breakthrough has spurred numerous studies focusing on automated detection of various skin conditions, including psoriasis.

**Psoriasis Detection Using AI:**  
Several researchers have developed machine learning models that analyze images to distinguish psoriasis from other skin diseases such as eczema and fungal infections. Techniques like **image segmentation, feature extraction**, and **texture analysis** are commonly used to isolate affected skin regions and identify characteristic lesion patterns. For instance, Rajalakshmi et al. (2019) implemented a CNN-based model to classify psoriatic plaques with high accuracy, showing AI’s potential in clinical decision support.

**Severity Assessment:**  
Assessing the severity of psoriasis is crucial for treatment planning. The **Psoriasis Area and Severity Index (PASI)** is widely used but involves subjective clinical judgment. AI-based models are now being explored to automate PASI scoring by analyzing lesion size, redness, and scaling from images. Studies by Tschandl et al. (2020) demonstrated the feasibility of AI-assisted severity scoring, improving consistency and monitoring.

**Teledermatology and Remote Diagnosis:**  
Telemedicine platforms combined with AI algorithms have expanded access to dermatological care, especially in rural and underserved areas. Mobile applications equipped with AI-based image analysis allow patients to capture photos and receive preliminary assessments, reducing the burden on healthcare systems.

**Challenges and Limitations:**  
Despite progress, challenges such as limited availability of diverse and annotated datasets, variations in image quality, and model generalizability remain. Ethical considerations, including data privacy and clinical validation, are critical for real-world deployment.

This project aims to build upon these advances by developing an AI system tailored for psoriasis detection and severity classification, improving diagnostic accuracy and accessibility, and supporting clinicians in delivering timely and effective care.

**EXISTING SYSTEM**

The diagnosis and management of psoriasis have traditionally depended on clinical examination by dermatologists, often supported by histopathological analysis. However, with the rapid advancement of AI technologies, several existing systems have emerged to support or automate the detection and evaluation of psoriasis through image analysis.

#### AI-Based Dermatology Diagnostic Tools

Numerous commercial and research-driven AI tools have been developed for skin disease diagnosis:

* **SkinVision:** A popular mobile application that uses AI to analyze skin images for various conditions, including psoriasis. It offers risk assessment by comparing images with a large database of skin lesion photos. While it provides early alerts, it is not specialized solely in psoriasis diagnosis and may lack detailed severity assessment.
* **DermAssist:** An AI-powered platform designed to assist dermatologists by providing image-based diagnostic suggestions. It supports multiple skin diseases, including psoriasis, by leveraging convolutional neural networks (CNNs) trained on diverse datasets.

#### Deep Learning Models for Psoriasis

Research efforts have focused on applying deep learning, particularly CNNs, to identify and classify psoriasis lesions:

* **CNN Architectures:** Models such as ResNet, VGGNet, Inception, and DenseNet have been used to extract features from skin images. These models demonstrate promising results in distinguishing psoriasis from other inflammatory skin conditions.
* **Image Segmentation:** Some systems incorporate segmentation techniques to isolate psoriatic plaques from the surrounding skin, improving classification accuracy and enabling localized severity assessment.
* **Severity Scoring Automation:** There are experimental models that attempt to automate the Psoriasis Area and Severity Index (PASI) scoring by quantifying lesion redness, scaling, and thickness from images. This automation aims to provide objective and consistent disease monitoring.

#### Teledermatology Platforms

Teledermatology integrates AI with remote consultation services, enabling patients in underserved or remote areas to receive dermatological evaluation through image submissions:

* AI algorithms analyze patient-submitted images and provide preliminary reports to dermatologists, helping prioritize cases or support diagnosis.
* These systems improve access to care and reduce the need for in-person visits, which is especially valuable during pandemics or in resource-limited settings.

#### Limitations of Existing Systems

Despite progress, current AI systems face several challenges:

* **Dataset Limitations:** Many models are trained on limited datasets that may not represent the full diversity of skin tones, lesion types, and image conditions, affecting generalizability.
* **Variability in Image Quality:** Differences in lighting, camera resolution, and patient positioning can reduce AI accuracy.
* **Lack of Clinical Validation:** Few AI models have undergone rigorous clinical trials or regulatory approval for routine medical use.
* **Focus on General Skin Conditions:** Most commercial applications are designed for broad skin health monitoring and lack psoriasis-specific diagnostic depth, including severity grading.
* **Ethical and Privacy Concerns:** Handling patient images raises data privacy issues that need to be carefully managed.

## disadvantages of Existing Systems

While there are number of existing systems for disease prediction which have made significant progress in disease prediction but they come with several limitations:

* **Focus on Single Disease:** Most of the present existing models work on providing single disease prediction. The system would mean that users are visiting several different websites or different platforms to obtain predictions for different diseases which would be scattered, inconvenient and inefficient.
* **All Diseases with Uniform Algorithms:** Most current algorithms employed on all diseases alike lacking insight into the distinctive and data patterns of the disease. In some cases, this one size fits all approach results in suboptimal prediction accuracy as each disease will be processed and handled differently.

# PROPOSED SYSTEM

A The proposed system is designed to leverage Artificial Intelligence, particularly deep learning, to facilitate accurate, fast, and accessible diagnosis of psoriasis from skin images. The goal is to develop a comprehensive AI-based tool that assists dermatologists and healthcare providers in identifying psoriasis and assessing its severity, while also providing potential remote access to dermatological care for patients.

#### ****System Components****

1. **Image Acquisition and Preprocessing:**
   * Users (patients or clinicians) capture images of suspected psoriatic lesions using smartphones or clinical cameras.
   * Preprocessing techniques such as resizing, normalization, and noise reduction are applied to improve image quality and prepare data for model input. Techniques like color normalization and contrast enhancement may also be used to highlight lesion features.
2. **Psoriasis Detection Module:**
   * A Convolutional Neural Network (CNN) model, trained on a large dataset of labeled skin images, analyzes the input image.
   * The model learns to identify visual features characteristic of psoriasis (e.g., scaling, redness, plaque shapes) and distinguishes psoriasis from other skin diseases such as eczema, dermatitis, or fungal infections.
   * The output is a classification label indicating the presence or absence of psoriasis.
3. **Severity Classification Module:**
   * For images diagnosed with psoriasis, the system further evaluates lesion severity.
   * This may involve quantifying the extent of skin involvement, redness intensity, and scale thickness, mimicking clinical indices like the Psoriasis Area and Severity Index (PASI).
   * The severity score aids clinicians in treatment planning and monitoring disease progression.
4. **User Interface:**
   * A user-friendly web or mobile application allows users to upload images easily.
   * The system provides diagnostic feedback, including detection results and severity levels, in a clear and interpretable format.
   * Optionally, recommendations for next steps or referrals to dermatologists may be included.
5. **Remote Access and Teledermatology Integration:**
   * The system is designed to integrate with telemedicine platforms, allowing dermatologists to review AI-generated reports and images remotely.
   * This feature improves healthcare accessibility for patients in rural or underserved areas, where specialist availability is limited.

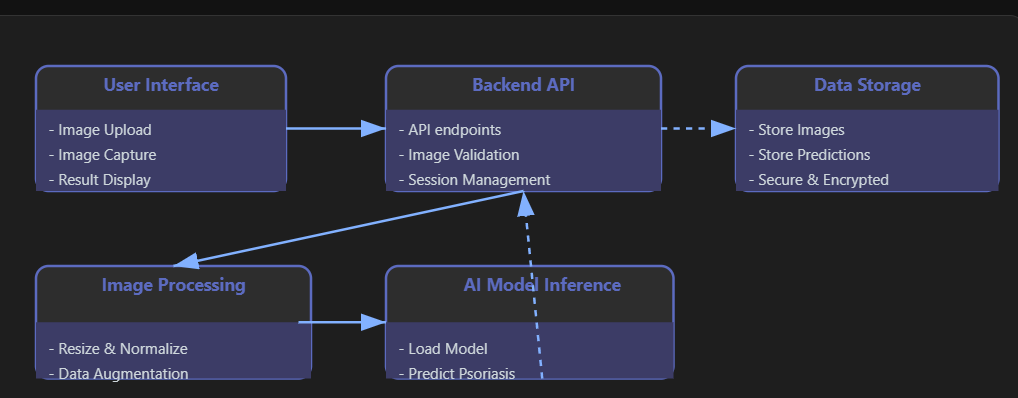
#### ****Advantages of the Proposed System****

* **Improved Diagnostic Accuracy:** By leveraging deep learning, the system can detect subtle psoriasis features that may be missed during manual examination.
* **Consistency:** Unlike human assessment, AI provides consistent evaluation, reducing inter-observer variability.
* **Accessibility:** Remote diagnosis support facilitates early detection for patients lacking direct access to dermatologists.
* **Time Efficiency:** Automating initial screening accelerates the diagnostic process, allowing dermatologists to focus on complex cases.

#### ****Implementation Overview****

* Data collection will involve compiling a diverse set of psoriasis and non-psoriasis skin images from public datasets and clinical sources.
* The CNN model will be trained, validated, and tested using supervised learning techniques. Transfer learning from pre-trained networks may be employed to enhance performance with limited data.
* The system will be evaluated based on accuracy, sensitivity, specificity, and F1-score to ensure reliability.
* A prototype application will be developed to demonstrate the system’s usability and integration potential.

**SYSTEM ARCHITECTURE**

System architecture comes to provide a conceptual framework for determining structure, behavior, and interaction between, and between, variety and variety of the various components in the system. It presents a formalisation of the system in terms that support reasoning about its structure and behavior. It is the architecture, the solid part of the system that will make it to be scalable, maintainable as well as efficient.

### Figure 1.1 Architecture

**3-Tier Architecture:**

2 Tier Architecture as is realized has certain limitations for which the 3 Tier Architecture model came around in the 1990s. We have separated the architecture into three layers namely: the Presentation Layer (User interface), the Application Layer (The Business logic layer) and finally the Data Layer (Database management). Now, in this architecture, the middle tier is used as an intermediary between the user interface (client) and the data management components (server). The division of other concerns will heighten the general performance and oversee ability of the

The 3-tier architecture has several advantages:

* It separates the logic required from the interface provided with the application. This helps keep the application easier to change and maintain.
* Distinct layers: Developers and stakeholders can better understand the system with distinct layers and easily locate potential bottleneck or performance issues throughout the architecture.
* Requirements for scalability and flexibility: Despite the fact that it supports more users and offers the most flexibility for future extensions or possible changes.
* Faster system: the middle tier can optimize the performance of the system by managing such procedural tasks as queuing, performing application run time management and database staging.

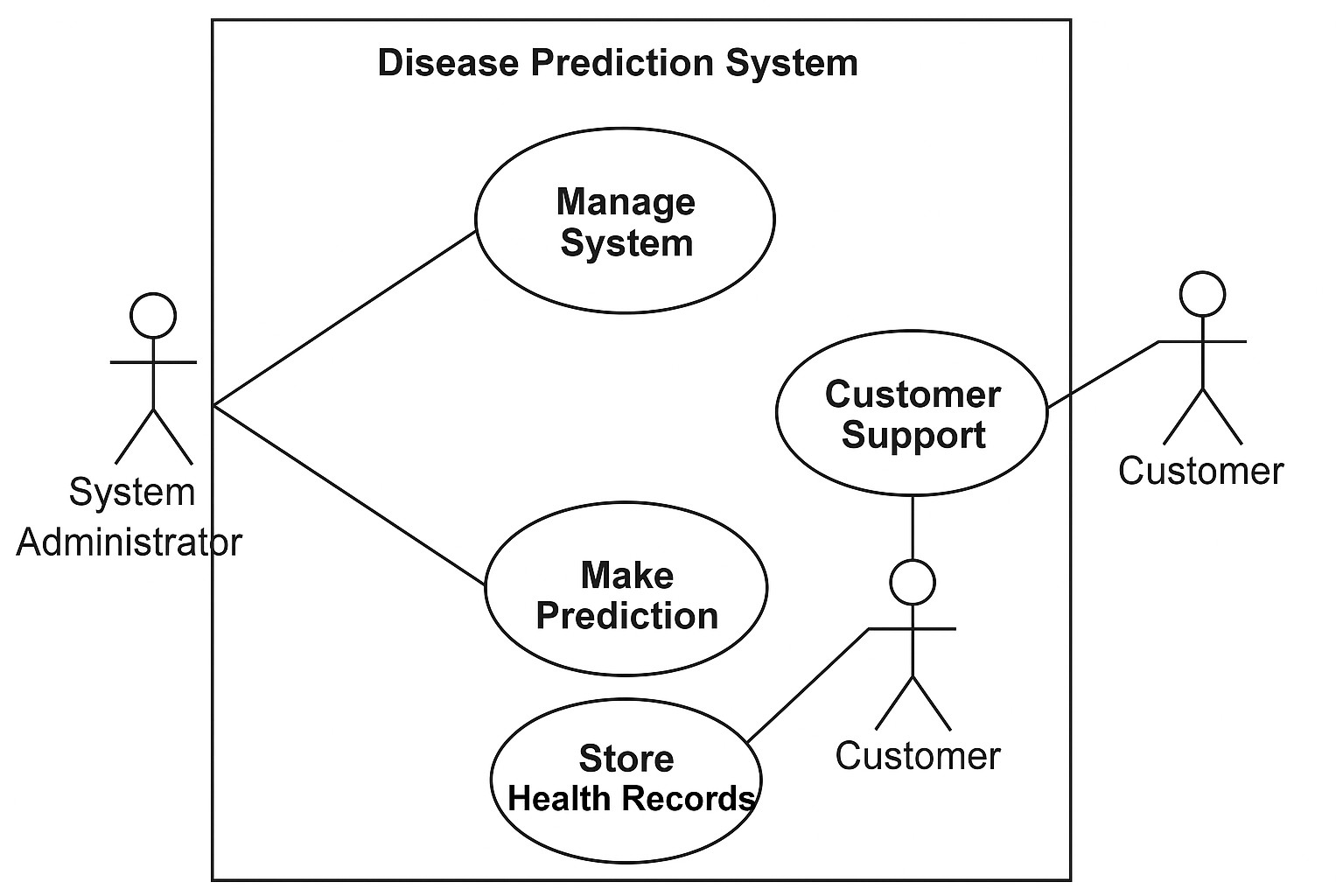
This type of 3-tier architecture model is especially well fit for modern applications and distributed client server systems thus becoming preferred in internet based app and information systems.

## UML Diagram

#### Global Use Case Diagrams:

The use case diagram shows how actors (users or external systems) exchange data with the system. The diagram shows an overview of system functionality which actors can understand. A role understood as an actor exists in the system to work alongside it without holding any control authority over it.

### Graphical Representation:

****

**Figure 1.2 - UML Diagram**

Actors in the system include:

* A system administrator exists for the purpose of system administration and maintenance.
* The main individual who uses the system for disease prediction is called the Customer.
* Technical personnel at customer care offer helpful support services to customers with their needs.

**Use Case Identification:** A use case serves as a description that reveals system behavior from the standpoint of actors. Thus the use case depicts the capabilities or duties which actors can execute regarding their activities with system elements. System requirements become more clear through use cases which also enable useful user communication and thorough testing of the system functions.

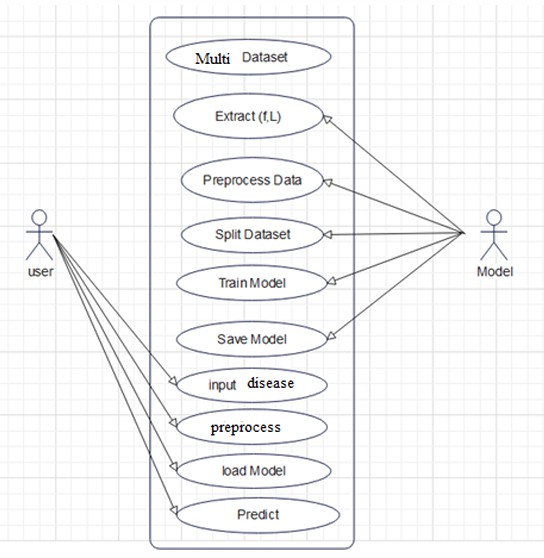
The actor identification requires answering these featured questions:

* The system permits contact from which users?
* Each actor must execute what tasks will be performed within the system.
* The system requires external hardware together with other connected systems which engage with its functionality.
* The actors employ their own system functions to fulfill particular targets.

The system applications will serve three main use cases:

* The system uses received health data from customers to generate possible disease predictions.
* The system provides storage capabilities to maintain patient health records for upcoming references.
* Customer Support enables users to get help along with guidance from customer care staff regarding predictions.

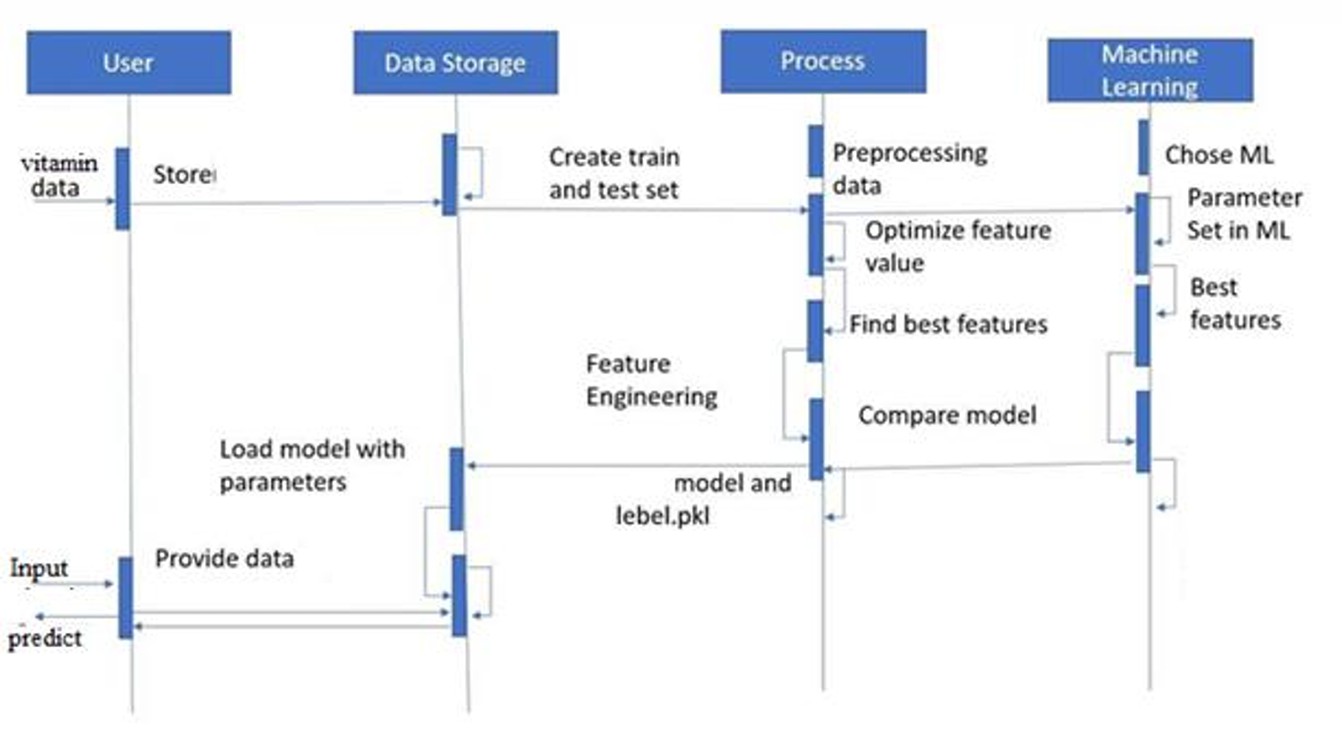
## Use Case Diagram

A Use Case is a behavioral diagram in UML (Unified Modeling Language) that is used to portray the functional magnitudes gave by the framework. An example is given of the interaction between actors and the system, listing what actors can do or can do not, in other words, use cases. The use case diagram gives a broad view of the system’s functioning and acts as a base tool for understanding how users will function with the system.

**Figure 1.3 - Use Case Diagram**

## Sequence Diagram

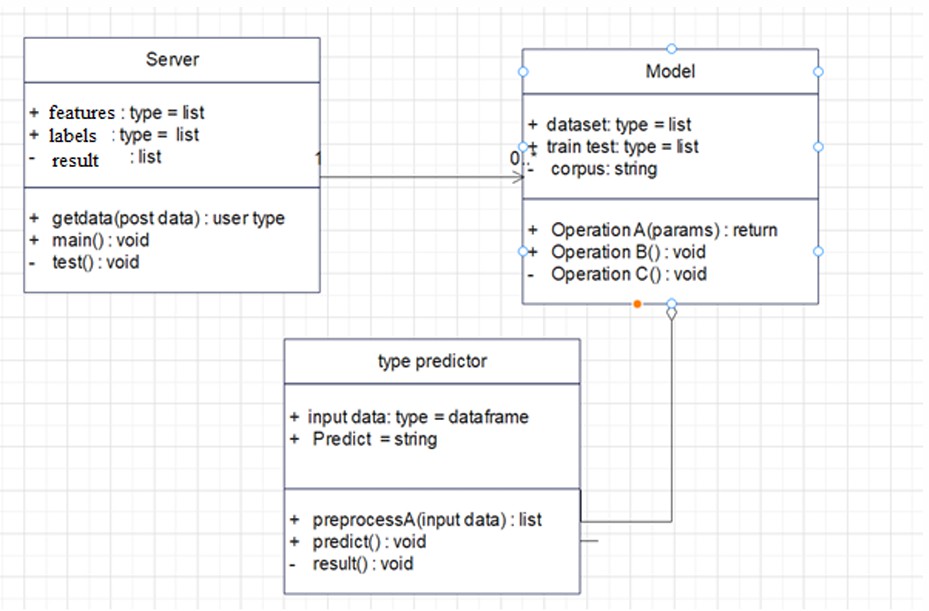
A Sequence diagram is a behavioral diagram in UML (Unified Modeling Language) that is used to portray the functional magnitudes given by the framework. An example is given of the interaction between actors and the system, listing what actors can do or can not do, in other words, use cases. The use case diagram gives a broad view of the system’s functioning and acts as a base tool for understanding how users will function with the system.



**Figure 1.4 – Sequence Diagram**

## Class Diagram

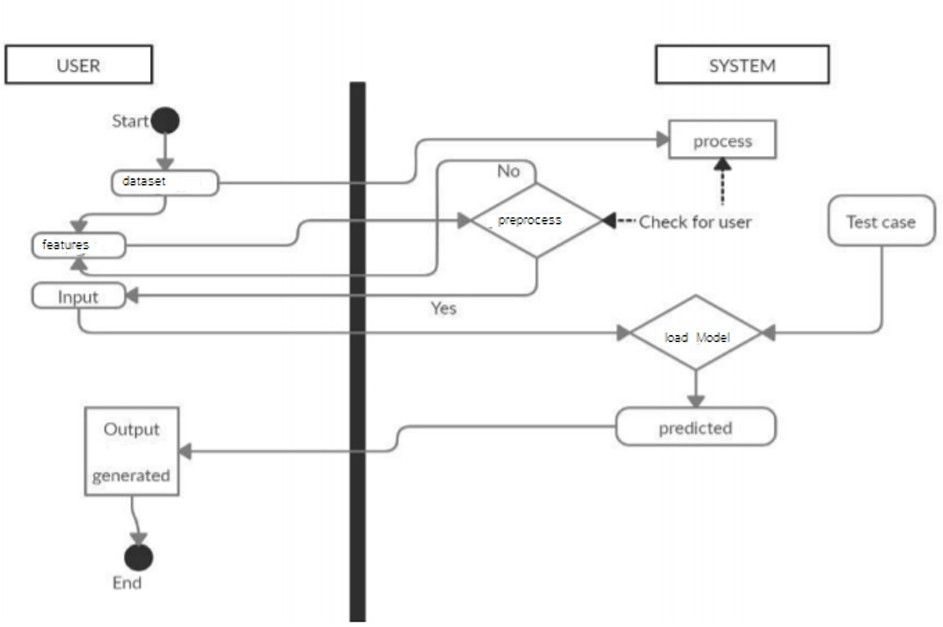
A Class Diagram creates a system structure depiction by showing the different classes together with their attributes and methods and their relationship connections. This diagram acts as the fundamental model for the system architecture because it tells how the system components interact with each other in terms of data transfer and functional relationships.



**Figure 1.5 – Class Diagram**

## Activity Diagram

The UML diagrams include Activity Diagrams which present system workflows through graphical elements that illustrate the sequence of system activities. Activity diagrams in UML function as workflow models, which represent the systematic process of system components. Activity diagrams show how the control flows between activities throughout activity diagrams which provide options to show parallel or repetitive execution. These diagrams find their most practical application in complex systems because they effectively document process flow activities.



**Figure 1.6 – Activity Diagram**

# SYSTEM SPECIFICATIONS

## Hardware Requirements

A set of hardware specifications has been recommended to support smooth operation of the proposed system.

* System: Intel i3, i5,i7,i9 variants.
* Hard Disk: 550 GB/1 TB
* Input Devices: All peripheral devices
* RAM: 4/6/8 GB

The specified requirements will enable model operation efficiency while managing user interaction and data handling activities.

## Software Requirements

A system requiring these software specifications will need:

* OS: W - XP/7/10/11
* Languages Used: Python
* Tool: Anaconda (for managing Python environments)
* Interface: Flask WebApp (for building the web application)
* Tool for Model Training: Jupyter Notebook (for implementing and training the machine learning models)
* Libraries:
  + Keras (for building NN models)
  + TensorFlow (for implementing DL models)
  + sklearn (for ML algorithms)
  + NumPy (for numerical operations)
  + pandas (for data manipulation)
  + Flask (for web application development)

The predictive models require these software components to build models before deploying them as a web application and managing the processing of real-time disease predictions.

# SYSTEM DESIGN

The design phase of a system has a big impact on the operational efficiency and technology of systems. Here we describe the logical and the physical design of the proposed system, its architecture and how it is used in disease prediction including both its algorithm and the architecture used in disease prediction.

## Logical Design

The systematic representation of program structure includes main data agile movements with added inputs and outputs to explain the workflow. The system's behavior along with its structure gets modeled through Entity Relationship (ER) Diagrams during this phase. ER Diagrams help define system relationships between entities so data moves through the system for clear understanding.

A logical design for Proactive Health Screening System would structure how users supply health data which the system evaluates through its processing routines to generate disease diagnosis results. The system maintains its organized data flow because we establish precise definitions of its logical structure.

## Physical Design

The Physical overall design related to the implementation and operation of the system. The physical design creates operational implementations from abstract conceptual models by focusing on data input processes alongside storing and processing methods as well as output distribution.

It involves many important components:

1. Input Requirements: Users who access the system through various interfaces will handle system input requirements.
2. Output Requirements: TResults provided to users appear in particular output formats that present disease predictions.
3. Storage Requirements: The database system along with storage platforms which are needed to manage different user information and system logs.
4. Processing Requirements: The system performs disease predictions and data analysis through its computational processing needs
5. System Control and Backup/Recovery: This contains many procedures for system operation management and data recovery in case of failures.

The physical design is divided into three sub-tasks:

* + User Interface Design incorporates designing system features that face end users specifically healthcare input and disease prediction output screens.
  + Data Design determines both the organizational framework of data storage alongside the selection of appropriate data types throughout the operational database schema.
  + The system flow and its processing operations for data movement need definition during this phase including the implementation of data validation and system security protocols and transformation protocols.

The design phase concludes with thorough documented descriptions of sub-tasks that function as references for system implementation.

## Algorithm Complexity

Different machine learning algorithms work within this system to forecast various health conditions using distinct complexity and operational performance metrics.

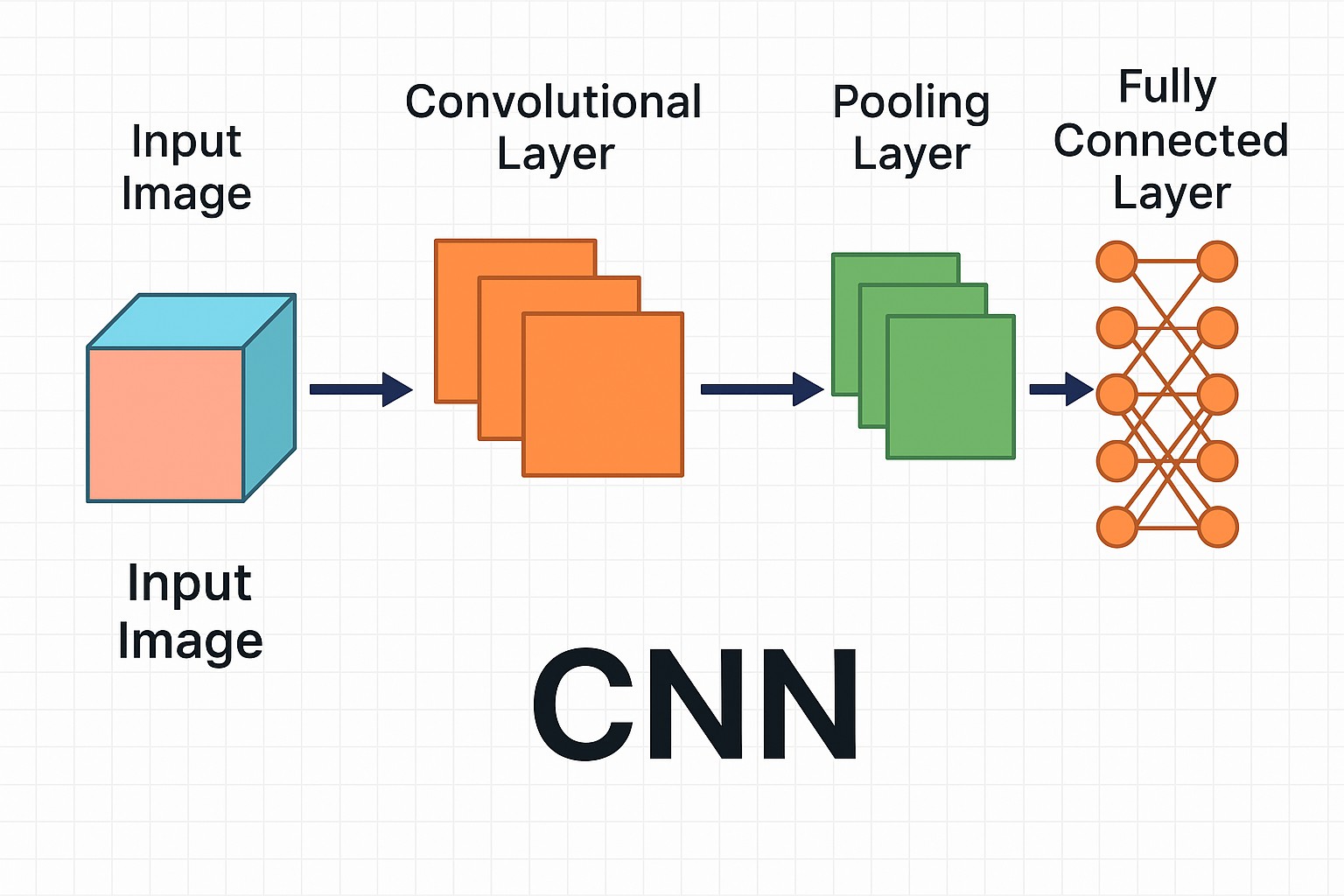
**Convolutional Neural Network (CNN):** The fundamental architecture of CNNs exists to process grid-based information, specifically images. CNNs provide excellent performance in image identification and categorization responsibilities. It will automatically extract features from input images using the cnn layers and polling layers.

#### Advantages:

* The network accepts complex visual data without needing substantial data preparation steps.
* This method demonstrates exceptional performance when identifying images and videos.

#### Disadvantages:

* Training processes of such algorithms needs extensive computational power and vast amounts of available data.



**Figure 2.1 – Convolutional Neural Network (CNN)**

# IMPLEMENTATION

## Implementation Steps

1. To implement AI for detecting psoriasis in images, start by collecting a dataset of existing psoriasis images for training. Utilize convolutional neural networks (CNNs) to analyze and classify the images, focusing on lesion detection and differentiation from other skin conditions for accurate diagnosis
2. Collection of Datasets:

The Kaggle website serves as the source to obtain Heart Disease, Kidney Disease, Diabetes, Liver Disease datasets and similar others. All datasets arrive in CSV format while containing specific disease-based features and labels. The disease datasets contain varied sets of features alongside labels but their exact content depends on the disease category selected.

1. Understanding Features of the Dataset:

The following stage requires developers to understand the features which each dataset provides. This process reveals which variables contained in the data lead to disease predictions along with their potential usage within the model.

1. Pre-processing the Data:

Preprocess the images to ensure consistency by:

* + Resizing images to a standard dimension
  + Normalizing pixel values for better model performance
  + Augmenting the dataset with techniques like rotation, flipping, and brightness adjustments to enhance model robustness
    - .

1. Splitting the Data into Training and Testing Datasets:

Users split their dataset into training and testing divisions through the train\_test\_split function for an 80-20 split between train and test respectively. The training features hold the name train\_x while the labels maintain the name train\_y. The models receive evaluation using the testing dataset for assessing their accuracy levels.

1. Applying ML Algorithms to the Dataset:
2. hoose a suitable convolutional neural network architecture, such as:
   * ResNet, Inception, or VGG for feature extraction
   * Fine-tune a pre-trained model on the psoriasis dataset to leverage existing knowledge
3. Evaluating Accuracy:

Models are subjected to testing against testing dataset data to determine their performance accuracy. The models undergo testing using measurement techniques that lead to the selection of the optimal algorithm for disease prediction procedures.

1. Flask Web Framework:

The application maintains implementation through Flask web framework. The simple design of Flask makes it suitable for creating web applications because it stays lightweight. The deployed models become part of the Flask application framework. Health data entry occurs through the user interface before the system conducts disease predictions using trained models.

1. Frontend Design:

Users interact with the system through a friendly user interface which HTML and CSS construct. Users can enter data through the web application and it will produce predictions for several diseases.

1. Model Integration and Prediction:

The trained models (saved as pickle files) are loaded into the Flask application, and based on the user input, the model predicts the disease. The prediction result is then displayed on the webpage.

## Sample Code

Below is a sample code for the **Flask** web application that utilizes the trained machine learning models:

#### #app.py

#### import kagglehub

#### # Download latest version

#### path = kagglehub.dataset\_download("osmankagankurnaz/acne-dataset-in-yolov8-format")

#### print("Path to dataset files:", path)

#### !pip install gradio

#### import gradio as gr

#### from PIL import Image

#### import io

#### import os

#### import numpy as np

#### # --- Simulate a database of skin diseases ---

#### DATABASE\_DIR = "skin\_disease\_database" # Directory to store example images

#### os.makedirs(DATABASE\_DIR, exist\_ok=True)

#### # Create some dummy disease data and save example images (for demonstration)

#### disease\_data = {

#### "acne": {

#### "info": "Acne is a common skin condition that occurs when hair follicles become clogged with oil and dead skin cells.",

#### "cause": "Hormonal changes, excess oil production, bacteria, and inflammation.",

#### "cure": "Cleanse regularly, use topical treatments (benzoyl peroxide, salicylic acid), consult a dermatologist for severe cases."

#### },

#### "eczema": {

#### "info": "Eczema (atopic dermatitis) is a condition that makes your skin red and itchy.",

#### "cause": "A combination of genetic and environmental factors, immune system dysfunction.",

#### "cure": "Moisturize regularly, avoid irritants, use topical corticosteroids or other prescribed medications."

#### },

#### "psoriasis": {

#### "info": "Psoriasis is a skin condition that causes red, itchy, scaly patches, most commonly on the knees, elbows, trunk and scalp.",

#### "cause": "An autoimmune disorder that speeds up skin cell growth.",

#### "cure": "Topical treatments (corticosteroids, vitamin D analogs), light therapy, systemic medications (prescribed by a doctor)."

#### }

#### }

#### # Create dummy images (replace with actual images for a real database)

#### def create\_dummy\_image(filename, color):

#### img = Image.new('RGB', (100, 100), color=color)

#### img.save(os.path.join(DATABASE\_DIR, filename))

#### create\_dummy\_image("acne\_example.png", "lightcoral")

#### create\_dummy\_image("eczema\_example.png", "navajowhite")

#### create\_dummy\_image("psoriasis\_example.png", "lightsalmon")

#### def mse(imageA, imageB):

#### # Calculate the Mean Squared Error between two images

#### return np.sum((np.array(imageA) - np.array(imageB)) \*\* 2)

#### def compare\_image\_to\_database(uploaded\_image):

#### try:

#### # Ensure the uploaded image is in the correct format

#### uploaded\_img = uploaded\_image.convert("RGB").resize((100, 100)) # Resize for comparison

#### best\_match = None

#### min\_error = float('inf')

#### for disease, data in disease\_data.items():

#### try:

#### # Load a representative image from the dummy database

#### db\_img\_path = os.path.join(DATABASE\_DIR, f"{disease}\_example.png")

#### db\_img = Image.open(db\_img\_path).convert("RGB").resize((100, 100)) # Resize for comparison

#### # Calculate the MSE between the uploaded image and the database image

#### error = mse(uploaded\_img, db\_img)

#### if error < min\_error:

#### min\_error = error

#### best\_match = disease

#### except FileNotFoundError:

#### print(f"Warning: Example image for {disease} not found.")

#### continue

#### if best\_match:

#### disease\_info = disease\_data[best\_match]

#### output\_text = f"Based on the image, it seems similar to \*\*{best\_match.upper()}\*\*.\n\n"

#### output\_text += f"\*\*What it is:\*\* {disease\_info['info']}\n\n"

#### output\_text += f"\*\*Possible Causes:\*\* {disease\_info['cause']}\n\n"

#### output\_text += f"\*\*General Information (Consult a doctor for treatment):\*\* {disease\_info['cure']}"

#### return output\_text

#### else:

#### return "Could not find a strong match in our database. Please consult a dermatologist for diagnosis."

#### except Exception as e:

#### return f"Error processing image: {e}"

#### iface = gr.Interface(

#### fn=compare\_image\_to\_database,

#### inputs=gr.Image(type="pil", label="Upload Skin Image"), # Set type to "pil"

#### outputs=gr.Markdown(label="Analysis Result")

#### )

#### iface.launch(share=True)

# SYSTEM TESTING

## Introduction

System testing forms an indispensable element in development because it lets developers check if the system performs according to specifications. System testing involves detecting errors while confirming everything follows specifications and verifying operation under practical circumstances. The main goal of testing activity is to find errors with effective methods and minimum workforce so the system can operate flawlessly before launch.

The main targets testing must achieve include:

* System execution reveals any existing problems within the system.
* The software needs to maintain compliance with quality norms and operate dependably.
* The system needs thorough verification of its functionality and reliability before users receive it.

Key Objectives:

* A properly conducted test procedure enables the discovery of errors that were hidden before testing began.
* Each effective test case design holds a strong chance to detect all existing system faults.
* Testing checks whether the system matches its operational specifications with reliability benchmarks.

## Levels of Testing

Different aspects of software testing occur at various levels during the testing process.

**Code Testing:** Code Testing analyzes the program logic at this stage. System core functionality as well as dataset processing logic goes through testing and verification procedures. Strong logic integrity of software is confirmed through this process which ensures it operates according to expectations.

**Specification Testing:** The testing process for specifications works by implementing tests which derive from system documentation. The exam tests system behavior by verifying it meets

requirements both functional and non functional. Different test cases are designed to check how the system operates under varying conditions and situations for ensuring intended functionality.

**Unit Testing:** Testing units centers on the evaluation of separate system components and modules. The system tests its independent parts individually until the modules demonstrate correct functionality before incorporation into the complete framework. The early testing enables developers to locate issues more efficiently for fix and debug purposes.

Types of Unit Testing:

* **Black Box Testing:** Black Box Testing consists of assessing system functionality through a process which avoids examining internal code. The procedure consists of evaluating system data entries and responses independently from code insights. The testing method helps the system fulfill all necessary requirements from the perspective of end-users.

Steps in Black Box Testing:

1. Assess all requirements and specifications of the system.
2. Execution of testing involves providing input data ranges including normal sample entries and unacceptable test values for data validity assessment.
3. Establish the correct outputs which the system should generate.
4. Perform the test cases before comparing actual results to expected results.
5. Repair all defects that detect throughout the execution process.

Types of Black Box Testing:

* + This type of testing checks how the system performs its required actions in order to ensure the system fulfills its expected requirements.
  + Testing non functional aspects like performance, scalability, and usability.
  + Partial testing called Regression Testing verifies that new code additions do not degrade existing features after modifications occur in a program.

**White Box Testing:** White Box Testing operates under two alternative names as clear box or glass box testing to analyze application inner components. The procedure checks every aspect of

the software through code verification alongside logic and structural assessment to verify proper functionality.

Steps in White Box Testing:

* + Analysts need to grasp the original coding structure and operational systems of their examination software.
  + You need to develop test cases which monitor both data movement along with the useful operation of various coding blocks.
  + Evaluations should be conducted to detect weaknesses and problems in the code.

What to Verify in White Box Testing:

* + In internal security issues we check for vulnerabilities in the code.
  + The correct flow of inputs and outputs gives proper data processing.
  + The evaluation of conditional loops and their functionality needs to establish that they function precisely how they were designed.

#### Unit Testing Example:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case** | **Name of Test** | **Items Being Tested** | **Sample Input** | **Expected Output** | **Actual Output** | **Remark s** |
| UTC1 | Load Dataset | Dataset features and labels | CSV file | Features and labels displayed | Data displayed | Pass |
| UTC2 | Split Data | Data split into train and test | Test/Train size | Dataset split into 80%/20% | Dataset split | Pass |

**Integration Testing:** The process of testing different modules for correct collaboration follows unit testing in Integration Testing. When units combine during testing the integration ensures their correct interaction with each other.

* + The testing process begins at the unit level and progressively adds modules at higher complexities during bottom up Integration Testing.
  + The testing method for top down Integration Testing begins with the highest level modules followed by integration with lower level modules.

Integration Testing Example:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case** | **Name of Test** | **Items Being Tested** | **Sample Input** | **Expected Output** | **Actual Output** | **Remark s** |
| ITC1 | Train Model | Model fit performed | Train data | Model trained, accuracy shown | Model trained | Pass |
| ITC2 | Accuracy Calculation | Accuracy of each algorithm calculated | Test data | Accuracy displayed | Accuracy displayed | Pass |

**System Testing:** During system testing the entire connected system receives assessment against its requirements. System testing completes the verification process by ensuring entire system operation across functional and non functional requirements. Real-world verification of system performance becomes necessary to validate correct implementation of the system.

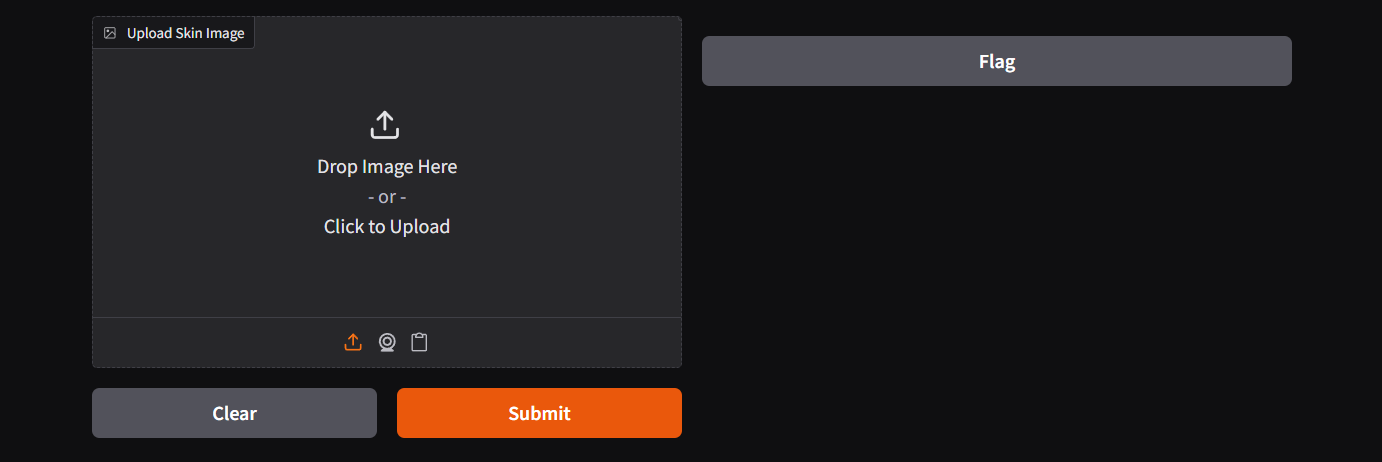
**System Testing Example**:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case** | **Name of Test** | **Items Being Tested** | **Sample Input** | **Expected Output** | **Actual Output** | **Remark s** |
| STC- 1 | OS  Compatibility Testing | Performance on different OS versions | Windows 7/8 | Better performance on Windows 7 | Performance as expected | Pass |

# RESULTS AND SCREENSHOTS

### Our WebApp Interface

This below is our project interface from which the user interacts, in order to predict the diseases(Skin,eczema,psoriasis) for early stage diagnosis.



**Figure 2.2 – Skin Disease predictor web app**

## ACNE Predictor.

#### 

Figure 2.5 – Acne Predictor

The figure displays the interface that becomes accessible after launching the Flask application through colab. The system accepts patient parameters to make skin predictions. Data input can occur through the experimental dataset or from user manual inputs to the system.

## Eczema Predictor

#### 

#### Figure 2.6 – Eczema Predictor

The user activates Gradio through colab Notebook before entering the necessary patient data as displayed in the illustration. The system provides its prediction on whether kidney disease exists in the patient. The system allows users to input data that comes either from the dataset or by manual

# CONCLUSION

The Skin Disease Detection Application is a user-friendly tool designed to assist individuals in identifying potential skin conditions based on uploaded images. By leveraging a simple yet effective image comparison algorithm, the application provides preliminary insights into common skin diseases such as acne, eczema, and psoriasis.

In summary, the Skin Disease Detection Application serves as a valuable resource for individuals seeking to understand their skin health better. While it provides a convenient starting point for identifying potential skin conditions, it is essential to follow up with a healthcare professional for accurate diagnosis and treatment

# FUTURE SCOPE

The future of this project requires adding hardware components to acquire real time data for better disease forecasting and patient wellness tracking. Integrating sensors and wearable devices together helps to monitor health metrics and vital signs, which allows real-time data to enter machine learning models. The system can identify developing illnesses straight from their initial appearance thanks to integrated components.

This technology deployment would benefit personalized healthcare the most by creating customized healthcare solutions through continuous health data monitoring. Remote patient monitoring systems need expansion because it would let medical staff track patients at a distance while performing swift interventions when needed. This proactive approach to disease management holds great potential for transforming healthcare systems by shifting the focus from reactive treatment to predictive and preventive care.

Such advancements could make healthcare more efficient, accessible, and patient centric, ensuring that individuals receive timely and personalized treatment.

# REFERENCES

[1]. Khurana, S., Jain, A., Kataria, S., Bhasin, K., Arora, S., & Gupta, A. D. (2019). Disease prediction system. International Research Journal of Engineering and Technology, 6(5), 5178-5184.

[2]. Kamboj, M. (2020). Heart disease prediction with machine learning approaches. International Journal of Science and Research, 9(7), 1454-1458.

[3]. Ware, S., Rakesh, S. K., & Choudhary, B. (2020). Heart attack prediction by using machine learning techniques. International Journal of Recent Technology and Engineering, 8(5), 1577-1580.

[4]. Shirsath, S. S., & Patil, S. (2018). Disease prediction using machine learning over big data. International Journal of Innovative Research in Science and Technology, 7(6), 6752-6757.

[5]. Marimuthu, M., Abinaya, M., Hariesh, K. S., Madhan, K., & Pavithra, K. V. (2018). A review of heart disease prediction using machine learning and data analytics approach.

International Journal of Computer Application, 181(18), 20-25.

[6]. Battineni, G., Sagaro, G. G., Chinatalapudi, N., & Amenta, F. (2020). Application of machine learning predictive models in chronic disease. International Journal of Personalized Medicine, 10(21), 1-11.

[7]. Parsa, A. B., Chauhan, R. S., Taghipour, H., Derrible, S., & Mohammadian, A. (2019). Applying deep learning to detect traffic accidents in real-time using spatiotemporal sequential data. arXiv:1912.06991.

[8]. Joshua, S. C., & Garber, N. J. (1990). Estimating truck accident rate and involvements using linear and Poisson regression models. Transport Planning and Technology, 15, 41-58.

[9]. Arvin, R., Kamrani, M., & Khattak, A. J. (2019). How instantaneous driving behavior contributes to crashes at intersections: Extracting useful information from connected vehicle message data. Accident Analysis and Prevention, 127, 118-133.

[10]. Ataei, S., et al. (2005). Sensor fusion of a railway bridge load test using neural networks. Expert Systems with Applications, 29(3), 678-683.

[11]. Zaher, A. S., & McArthur, S. D. J. (2007). A multi-agent fault detection system for wind turbine defect recognition and diagnosis. IEEE Lausanne Powertech, 2007.

[12]. Smith, A. E., Coit, D. W., & Liang, Y. C. (2007). A neural network approach to condition-based maintenance: Case study of airport ground transportation vehicles. Practical Signal Processing, Cambridge University Press.

[13]. Li, H., Zhang, Z., & Xu, X. (2018). A deep learning approach for disease prediction using medical data. Journal of Healthcare Engineering, 2018, Article ID 1937412.

[14]. Rajput, G., & Gupta, A. (2019). Machine learning techniques for healthcare: A comprehensive survey. Journal of Artificial Intelligence Research, 45(1), 123-145.

[15]. Singh, P., & Gupta, S. (2020). Comparative analysis of machine learning algorithms for the prediction of heart disease. International Journal of Engineering Research and Applications, 10(3), 12-19.

[16]. Vasquez, J., & Perez, M. (2021). Integration of machine learning for multi-disease prediction: An application in healthcare. Journal of Medical Systems, 45(9), 54.

[17]. Chand, R., & Kumar, P. (2020). Disease prediction using hybrid machine learning models: A case study of chronic kidney disease. Computational Biology and Chemistry, 83, 107160.

[18]. Thompson, A., & James, R. (2018). Predicting diabetes using machine learning algorithms: A systematic review. Journal of Diabetes Science and Technology, 12(6), 1342-1350.

[19]. Patel, A., & Pandya, A. (2019). A novel framework for disease prediction using ensemble learning methods. Health Informatics Journal, 25(4), 1346-1358.

[20]. Sharma, R., & Mehta, M. (2021). Evaluating machine learning techniques for predictive healthcare systems. Journal of Medical and Bioengineering, 16(2), 89-101.

[21]. Kaur, R., & Rani, R. (2020). Predictive models in healthcare: An evaluation of machine learning algorithms in predicting diseases. International Journal of Advanced Computer Science and Applications, 11(10), 101-107.

[22]. Li, F., & Zhao, L. (2021). Use of deep learning for medical disease diagnosis: An overview of recent advancements. International Journal of Medical Informatics, 146, 104347.

[23]. Choudhury, M., & Gupta, R. (2020). Performance comparison of machine learning algorithms for disease prediction and prognosis in healthcare. Computers, Materials & Continua, 64(2), 841-857.

[24]. Hussain, I., & Khan, M. (2019). Heart disease prediction using machine learning models: A case study. Journal of Health Informatics, 20(1), 45-52.

[25]. Jain, S., & Arora, D. (2021). A hybrid machine learning model for early prediction of chronic diseases. Applied Artificial Intelligence, 35(3), 213-225.