



**ÇANKAYA UNIVERSITY**  
**FACULTY OF ENGINEERING**  
**COMPUTER ENGINEERING DEPARTMENT**

**CENG 407**

Innovative System Design and Development I

Project Report

**TEAM ID: 202402**

**Skinalyzer**

202011039 Bekir Emrehan Şimşek

202011013 Melike Hazal Özcan

202011407 Bilgesu Fındık

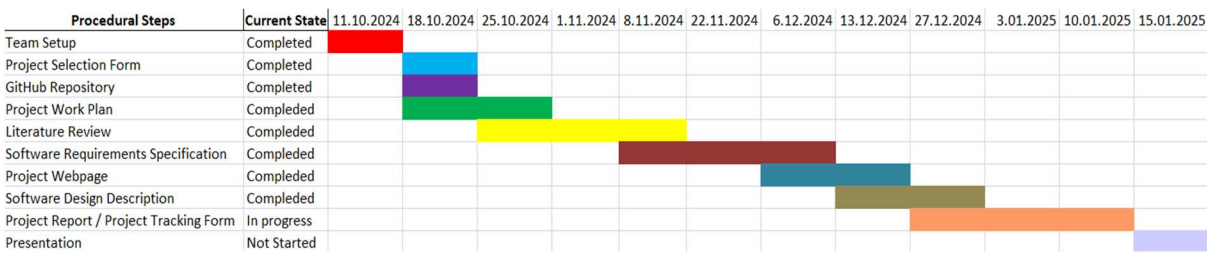
202011048 Pelin Koz

Advisor: DR. Ayşe Nurdan SARAN

## Introduction

The **Skinalyzer: AI-Based Skin Cancer Detection System** is an innovative desktop application designed to facilitate the early detection of skin cancer through advanced machine learning techniques, including Federated Learning (FL) and Incremental Learning (IL). The system empowers users—ranging from healthcare professionals to individuals—by allowing them to upload skin lesion images for analysis while prioritizing data privacy through local processing. Using FL, updates from multiple devices are aggregated to enhance a global model without compromising user data. Incremental Learning ensures the system adapts continuously to new data, retaining prior knowledge for improved diagnostic accuracy. By integrating state-of-the-art AI techniques with a user-friendly interface, Skinalyzer addresses challenges in data heterogeneity and privacy, offering a scalable, secure, and effective solution for skin cancer diagnosis.

## Project Work Plan



## Litreture Review

## Abstract

This project aims to develop a more sensitive and scalable machine learning model by combining incremental learning and federated learning techniques, focusing on skin cancer detection. Incremental learning allows the model to increase its accuracy as new data streams are integrated; thus, the model is updated as new images are added, improving its performance. Federated learning, on the other hand, enables processing in a distributed manner without transferring the data to a central server, protecting user privacy and allowing the model to be updated on various devices. The ISIC 2019 Skin Lesion Images for Classification and HAM10000 datasets used in our study provide a large sample of skin lesions that help the model recognize different types of skin cancer. These methods ensure that the model is privacy-aware and continuously updatable, while encouraging the effective use of AI-supported diagnostic systems in the healthcare

field. As a result, this project aims to make an innovative contribution to personalized and sensitive diagnoses by integrating dynamic and reliable AI solutions into healthcare technologies.

## **Introduction**

In recent years, advances in the field of artificial intelligence have caused radical changes in many areas, especially in the healthcare sector. Artificial intelligence technology enables diseases to be diagnosed quickly and with high accuracy rates by utilizing big data. Artificial intelligence algorithms, especially in the detection of abnormalities such as tumors, lesions, and aneurysms, enable images to be evaluated more quickly and precisely. In our graduation project, we plan to use incremental learning and federated learning techniques in skin cancer diagnosis; because incremental learning increases diagnostic accuracy by allowing the model to be constantly updated, while federated learning protects patient privacy by enabling transactions between devices without transferring data to a central server. Processing of image data enables detailed analysis of skin lesions, contributing to the accurate recognition of different types of skin cancer, and supports the training of the model with a wide range of data. These two methods allow the model to be trained on various datasets while preserving the confidentiality of the data, thus strengthening its capacity to recognize different types of skin cancer and providing a more scalable structure.

## **Incremental Learning**

Incremental Learning is a machine learning technique that enables a model to learn continuously by incorporating new data over time without retraining from scratch. Unlike traditional methods where models are trained on a fixed dataset, incremental learning allows the model to update its knowledge dynamically as new information becomes available. This approach is particularly useful in scenarios where data is constantly evolving, such as medical diagnostics, where new patient data can improve the model's accuracy and adaptability. By retaining prior knowledge while learning from new data, incremental learning ensures that the model remains current, effective, and efficient, making it highly valuable for applications that require continuous improvement. Incremental learning is effectively used with image datasets, where new visual data is continuously added. In medical imaging, for example, this approach allows the model to learn new patterns from incoming images without forgetting prior knowledge, thus improving diagnostic accuracy over time. By progressively incorporating new image data, incremental learning enhances the model's ability to handle

diverse and dynamic visual information, making it ideal for applications where data is frequently updated.

**There are some academic studies using incremental learning methods on ISIC2019 and HAM10000 datasets and other image datasets.**

Study Title	CNN Model	Metrics	Incremental Model
Development and Validation of Adaptable Skin Cancer Classification System Using Dynamically Expandable Representation [1]	Expandable CNN with convolutional layers, batch normalization, dropout, ReLU, and softmax for classification.	Accuracy: 80.88%, Weighted-Average Precision:91.9%, Weighted-Average Recall: 80.9%, Weighted-Average F1- Score: 84.8%, AUC: 94.3% (HAM10000), External Validation AUC: 91.1% (ISIC 2019)	Dynamic Architecture, where only the last layers are modified for new classes.
AMIAC: Adaptive Medical Image Analysis and Classification Framework [2]	CNN with batch normalization	Accuracy: 97.811%, F1 Score: 97.35%, Precision:97.19%, Efficiency: High	Adaptive Learning combined with incremental learning, which uses batch normalization and dropout adjustments for new classes.
BLSNet: Skin Lesion Detection and Classification Using Broad Learning System with Incremental Learning Algorithm [3]	Broad Learning System with CNN backbone for feature extraction.	Accuracy: 99.09%, F1 Score: 98.73%, Execution Time: 0.93 seconds	Broad Learning with incremental node addition, which expands the network architecture without retraining.
An Appraisal of Incremental Learning Methods [4]	Progressive Neural Networks (PNN)	EWC Accuracy: 94.5 % on permuted MNIST,  SI Accuracy: 97.5% on MNIST,  ICaRL Accuracy: around 62.5 – 67 % on CIFAR-100	Elastic Weight Consolidation (EWC), iCaRL (Incremental Classifier and Representation Learning), SI (Synaptic Intelligence), PNN
Comparing Incremental Learning Strategies for Convolutional Neural Networks [5]	Convolutional Neural Networks (CNN)	CaffeNet + Svm Accuracy: from 41.63% to 66.97% (iClubWorld28), CaffeNet + Ft Accuracy: 78.4%, Vgg_Face+SVM Accuracy: 96.73%	LeNet7: A seven-layer CNN,  CaffeNet + SVM,  CaffeNet + Fine Tuning (FT), VGG_Face + SVM

The first academic study on the table: **Development and Validation of Adaptable Skin Cancer Classification System Using Dynamically Expandable Representation** aimed to develop a scalable skin cancer classification model using the Dynamically Expandable

Representation (DER) algorithm with incremental learning to enhance diagnostic accuracy and adaptability. Trained on the HAM10000 and ISIC 2019 datasets, the

model progressively expands its classification capacity to recognize a wide range of skin lesions. Based on the ResNet-50 architecture, the model uses data augmentation techniques, including horizontal flipping, contrast-brightness adjustments, and distortion, to increase robustness against variations. Achieving a weighted-average precision of 0.918, recall of 0.808, F1-score of 0.847, and an AUC of 0.943 on the HAM10000 dataset, the DER model outperformed traditional methods in accuracy and adaptability, with an AUC of 0.911 on the ISIC 2019 validation. This incremental learning approach successfully mitigates catastrophic forgetting, enabling the model to retain knowledge from prior classes while effectively integrating new data.

Second study: **AMIAC** presents an adaptive self-learning framework for medical image analysis, allowing deep learning models to adapt to changing image distributions over time. The framework combines manual and CNN-based feature extraction, enhancing model accuracy and reducing the need for retraining. Manual features capture specific image details, while pre-trained CNNs extract abstract features, providing complementary information.

Trained initially on a large dataset, the model adapts to new imaging modalities through transfer and incremental learning. The framework achieved high performance, with an F1-score of  $97.35 \pm 0.39$ , precision of  $97.19 \pm 0.51$ , and accuracy of  $97.81 \pm 0.35$ , indicating its potential as a tool for supporting pathologists in diagnosis.

### **BLSNet: Skin Lesion Detection and Classification Using Broad Learning System with Incremental Learning Algorithm:**

The BLSNet model in this paper is based on a Broad Learning System (BLS) architecture, which emphasizes wide feature representations rather than deep stacking of layers. This approach is combined with incremental learning techniques, allowing the model to learn new data classes without retraining on the entire dataset, a critical feature in handling continuously updated medical image datasets. For feature extraction, BLSNet uses a combination of manual and CNN-based features, where the manual features capture specific skin lesion characteristics, while the CNN-based features are extracted from pretrained models. To further enhance the robustness of the model, data augmentation techniques are applied, including horizontal flips, random contrast and brightness adjustments, and distortion to simulate variations in real-world image conditions. This allows the model to generalize better across different imaging conditions.

During training, an auxiliary loss function is applied to balance knowledge from previously learned and new classes, thus mitigating catastrophic forgetting. The BLSNet achieves high performance on ISIC 2019 and PH2 datasets, with an accuracy of 99.09% and an F1-score of 98.73%, as well as a low processing time of 0.93 seconds per image, making it both precise and efficient.

## **Federated Learning**

Federated learning is a method that enables machine learning models to be trained on distributed data located on different devices or systems without the need to centralize the data on a server. In this model, each device or system trains the model on its local data and only sends the model parameters to the central server. This approach prevents data

from leaving the device, thus preserving data privacy, while also significantly reducing network traffic. The central server combines the parameters from devices to update the model and then sends the updated model back to the devices. This iterative process allows the model to continuously improve without data ever leaving the devices. Federated learning offers a major advantage in areas with sensitive data, such as healthcare, by enhancing security in medical applications and protecting patient privacy; for example, electronic health records, including lab results, biomedical images, and vaccination status, can be processed as primary health data sources for machine learning applications without leaving the device.[6] In this way, data privacy is preserved, and powerful, secure, and continuously learning models are developed by staying up to date with local data on devices.

Study Title	CNN Model	Metrics	Federated Model
Federated Machine Learning for Skin Lesion Diagnosis: An Asynchronous and Weighted Approach [7]	CNN and CNN components Convolutional layers, Max- pooling layers, Fully connected layers, Dropout regularization	F1: 94.8, Sensitivity: 94.1, Recall: 92.6, Specify: 96.3, Loss: 1.6, Precision 96.7, Accuracy: %92	Asynchronous federated learning (Async-FL)
Federated and Transfer Learning Methods for the Classification of Melanoma and Nonmelanoma Skin Cancers: A Prospective Study [8]	CNN and CNN architectures, including ResNet, VGG, DenseNet, GoogLeNet, and MobileNet.	AUC: 99.43% Accuracy: 94.17% Recall rate: 93.76% Precision: 94.28% F1-score: 93.93%	Transfer learning , fine-tuning and semi-supervised FL framework motivated by a peer learning (PL)
Deep Learning Espoused Imaging Modalities for Skin Cancer Diagnosis: A Review [9]	Generative Adversarial Network (G-AN)	Accuracy: %98.5 , Sensevity: % 95, Specify: %92, Dice Coefficient: %96	Federated Learning (FL)
An Adaptive Federated Machine Learning-Based Intelligent System for Skin Disease Detection: A Step toward an Intelligent Dermoscopy Device [10]	Convolutional Neural Networks (CNNs) Single-instance optimized CNN model Dynamic ensemble classifiers	Accuracy: %95.6, Recall:%95, Precision :%95	Online training (OT), Federated Learning, Online classifier updating (OCU)

### **Federated Machine Learning for Skin Lesion Diagnosis: An Asynchronous and Weighted Approach**

The first article on the table focuses on enhancing skin cancer detection while preserving privacy through various techniques: asynchronous federated learning (Async-FL) for efficient model updates, convolutional neural networks (CNNs) for feature extraction, layer-by-layer asynchronous updates to minimize communication load, dropout regularization to reduce overfitting, and a client selection algorithm to optimize accuracy

based on data and hardware. This combination aims to improve diagnostic performance effectively. The study's results using asynchronous federated learning (Async-FL) and convolutional neural networks (CNNs), with an F1 score of 94.8, sensitivity of 94.1, recall of 92.6, specificity of 96.3, precision of 96.7, and accuracy of 92%. The model's loss was recorded at 1.6, indicating strong performance in skin cancer detection while maintaining privacy and efficiency.

### **Federated and Transfer Learning Methods for the Classification of Melanoma and Nonmelanoma Skin Cancers: A Prospective Study**

The study investigates various federated learning (FL) and transfer learning (TL) techniques for classifying melanoma and nonmelanoma skin cancer. It emphasizes ensemble methods that integrate multiple CNN architectures, such as ResNet, VGG, and DenseNet, which significantly improve classification accuracy. Notably, one study achieved a remarkable area under the curve (AUC) of 99.43%, with an accuracy of 94.17%, recall of 93.76%, precision of 94.28%, and an F1-score of 93.93%. This review synthesizes findings from a systematic search across benchmark datasets, including HAM10000, to highlight recent FL and TL algorithms for early-stage treatment.

### **Deep Learning Espoused Imaging Modalities for Skin Cancer Diagnosis: A Review**

The objective of this article is to deliver a thorough examination of the progress, obstacles, and prospective applications within this vital area of dermatology. In this study, various neural network architectures were evaluated for skin cancer detection, including Generative Adversarial Network (G-AN), Artificial Neural Network (A-NN), Convolutional Neural Network (C-NN), and Kernel-based SNN (KSNN). Among these, the G-AN achieved the best performance, yielding an Area Under the Curve (AUC) of 98.5%, a sensitivity of 95%, a specificity of 92%, and a Dice Coefficient (DC) of 96%. This highlights the G-AN's superior capability in accurately detecting skin cancer lesions.

### **An Adaptive Federated Machine Learning-Based Intelligent System for Skin Disease Detection: A Step toward an Intelligent Dermoscopy Device**

The article proposes an adaptive federated learning model designed for skin disease diagnosis, centering on an ensemble CNN as the main classifier to support intelligent dermoscopy applications. A single instance optimized CNN model is incorporated within the cloud server's ensemble to enhance adaptability. Key contributions include model deployment across both cloud servers and edge devices, facilitating continuous updates through online

training (OT) and online classifier updating (OCU). Using TensorFlow Federated and Federated Core APIs, this approach provides robust performance while adapting to new

disease data. Results indicate an accuracy of 95.6%, recall of 95%, and precision of 95%, demonstrating the model's effective and reliable diagnostic capability.

Paper Name	Image Processing	Metrics	Incremental Model	Federated Learning
Incremental learning and Federates Learning for Heterogeneous Medical Image Analysis	Traditional Deep Learning and CNN	<ul style="list-style-type: none"> <li>– CCSI accuracy: 79.10% (BloodMnist Dataset)</li> <li>– FeImpres (Medical init): 94.2% (<math>\alpha=0.01</math> epoch=5 BloodMNIST Dataset)</li> <li>– FedImpres w constrained CE loss: 80.6% (<math>\alpha=0.005</math>, epoch=10, Retina Dataset)</li> </ul>	Continual Normalization model structure to restore Continual Class-Specific Impression	FedCurv for Catastrophic Forgetting, Transfer Learning, FedImpres

The thesis[11] aims to improve the adaptability of deep learning (DL) models in real-world settings, where data is often heterogeneous, making it crucial to enhance these models' practical utility. A combined approach using incremental and federated learning, as discussed in the paper "*Incremental Learning and Federated Learning for Heterogeneous Medical Image Analysis*," strategically addresses two major challenges in medical imaging: the need for continuous learning with new data and the requirement for data privacy across multiple institutions. This dual approach not only enables DL models to evolve with new disease classes but also ensures that data remains localized, preserving patient confidentiality.

By combining incremental and federated learning, this methodology offers a robust solution to meet evolving healthcare demands. Incremental learning combats "catastrophic forgetting," allowing the model to retain knowledge of prior disease classes while adapting to new ones. Federated learning, on the other hand, facilitates collaborative model training across institutions without centralizing sensitive data, thereby enhancing privacy. Together, these methods allow the model to generalize effectively across diverse datasets from various institutions, addressing "domain shift" challenges stemming from differing imaging protocols. This approach also minimizes the need for extensive data transfers and retraining, making it resource-efficient and adaptable to institutions with varied computational resources.

However, implementing both frameworks together is not without challenges. It demands precise model updates and coordination across clients, adding complexity to the system. Additionally, while synthetic data is valuable for privacy preservation, it may not fully capture real-world distributions, potentially reducing model accuracy over time if not carefully calibrated. These limitations underscore the need for rigorous model tuning and effective communication between institutions to optimize the potential of this advanced approach.



## Contribution

This project aims to advance the application of artificial intelligence in skin cancer detection by combining incremental and federated learning techniques on the ISIC2019 and HAM10000 datasets. This hybrid approach addresses two critical needs in medical AI: adaptability and privacy. Incremental learning allows our model to dynamically integrate new data over time, continuously enhancing diagnostic accuracy without the need for retraining and without compromising performance metrics such as accuracy. Federated learning, on the other hand, ensures that sensitive patient data remains decentralized by training across distributed devices without transferring information to a central server, thereby promoting data privacy.

Using both the ISIC2019 and HAM10000 datasets, our model will gain exposure to a wide range of skin lesion images, enhancing its ability to generalize across different types of skin cancer. This approach not only improves the model's scalability and privacy but also contributes to a more personalized, up-to-date, and secure diagnostic solution in dermatology. Through this innovative combination, we aim to set a new standard in real-world healthcare AI applications, where continuous learning, privacy preservation, and stable accuracy are critical.

In the incremental learning phase, we will use the **novel data-free class-incremental learning approach** combined with **Continual Class-Specific Impression (CCSI)**, implemented in **PyTorch**. This combination enables the addition of new classes without accessing old data, effectively minimizing knowledge loss while preserving class-specific impressions in the model's memory. To further enhance the approach, we will integrate **Flower** for federated learning. The data privacy and distributed learning capabilities of federated learning, combined with our incremental learning strategies, allow each device to learn using only its own data while retaining previous class impressions, thereby strengthening the model's long-term memory and overall performance.

# Software Requirements Specification

- **INTRODUCTION**

- **Purpose**

This document provides a detailed specification of the requirements for a skin cancer detection system. Combining Federated Learning and Incremental Learning techniques, the project aims to assist users in analyzing skin lesions to enable early detection of cancer risk. It serves as a guide for the development, implementation, and testing of the system.

- **Scope of Project**

This project aims to develop a desktop application that supports skin cancer diagnosis by analyzing skin lesion images. The system processes the images uploaded by users through advanced machine learning models, providing diagnostic results in a clear, accessible, and actionable format. By integrating Federated Learning, the system ensures the privacy of sensitive user data, as all raw data remains on local devices and is never transferred to a central server. This privacy-preserving approach aligns with modern data protection standards and user expectations in the healthcare sector.

Additionally, the system leverages Incremental Learning to continuously update the machine learning model with new data while preserving knowledge from previous datasets. This capability ensures that the model adapts to evolving data distributions without losing prior learning, maintaining high performance and accuracy over time. The combination of Federated Learning and Incremental Learning creates a scalable and secure framework for collaborative model training across multiple devices, addressing the challenges of data heterogeneity and privacy in medical applications.

Designed for healthcare professionals, researchers, and even individual users, the application aims to streamline the diagnostic process by offering a user-friendly interface and actionable insights. By training the model on diverse datasets like ISIC 2019 and HAM10000, the system ensures robust performance across various skin cancer types. This innovative solution has the potential to revolutionize early cancer detection by making cutting-edge AI technology accessible, secure, and reliable for clinical and personal use.

- **Glossary**

- **Skin Lesion:** An abnormal area of skin that may indicate a skin condition or disease, including potential cancerous growths.

- **Machine Learning (ML):** A branch of artificial intelligence that enables systems to learn and improve from data without being explicitly programmed.
- **Federated Learning:** A privacy-preserving machine learning method where the model is trained across multiple devices or servers without sharing raw data.
- **Incremental Learning:** A machine learning approach that allows a model to continuously update with new data while retaining knowledge from previous datasets.
- **ISIC 2019 Dataset:** A dataset containing labeled skin lesion images used for research and training in skin cancer detection.
- **HAM10000 Dataset:** A large collection of dermatological images used for diagnosing various skin conditions, including cancer.
- **Client Device:** A computer or device used by participants (e.g., healthcare professionals) to perform local training and interact with the system.
- **Central Server:** A server that aggregates updates from multiple client devices to improve the global machine learning model.
- **Model Update:** The process of incorporating new training data into an existing machine learning model to improve its accuracy and performance.
- **Neural Network:** A computational model inspired by the human brain, used in machine learning to identify patterns and make predictions.
- **Confidence Score:** A metric that indicates the certainty of the model's prediction, often expressed as a percentage.
- **Python:** A programming language commonly used for developing machine learning and AI applications.
- **PyTorch:** An open-source machine learning library widely used for building and training neural networks.
- **Tkinter:** A Python library used for creating graphical user interfaces (GUIs).
- **PyQt5:** A set of Python bindings for the Qt application framework, used to develop advanced GUIs.
- **Kafka-ML:** An open-source tool that facilitates communication and data exchange in federated learning systems.
- **HTTPS:** A secure protocol for communication over the internet, ensuring data encryption between clients and servers.

- **TLS/SSL:** Encryption protocols used to secure internet communications, ensuring data integrity and privacy.
- **Data Preprocessing:** The process of preparing raw data (e.g., skin lesion images) for machine learning by cleaning, resizing, and formatting it appropriately.
- **Error Logging:** The practice of recording errors or failures in a system to facilitate debugging and system improvement.
- **Graphical User Interface (GUI):** A visual interface that allows users to interact with a system using graphical elements such as buttons and menus.
- **Probability Analysis:** The process of estimating the likelihood of an outcome, such as determining whether a skin lesion is cancerous.
- **Data Integrity:** Ensuring that data is accurate, consistent, and free from unauthorized modifications during transmission or storage.
- **Global Model:** A centralized machine learning model that aggregates updates from client devices in a federated learning system.

- **Overview of the Document**

This document provides a comprehensive specification of the requirements for the skin cancer detection system, ensuring a clear understanding of the project's objectives, functionality, and technical expectations. It begins with an introduction outlining the purpose, scope, and goals of the project, accompanied by relevant references and a glossary of key terms. Following this, the overall description section highlights the primary features of the system, its intended users, and its dependencies on external technologies. The core of the document is the requirements specification, which details both functional requirements, such as the system's ability to analyze images and provide diagnostic insights, and non-functional requirements, including performance, usability, and security considerations. Additionally, the document describes the external interfaces, outlining interactions with hardware, software, and communication protocols, while also emphasizing critical system attributes such as portability, scalability, and adaptability. Finally, the references section provides a curated list of datasets, research articles, and external resources that inform the system's design and development. This document serves as a foundational guide for developers, testers, and stakeholders, supporting all phases of the project from design to deployment.

## 2. OVERALL DESCRIPTION

- **Product Perspective**

This system represents an innovative approach to medical diagnostics by combining image analysis with artificial intelligence. It aims to make advanced diagnostic tools accessible to

general users, ensuring simplicity and usability. The system addresses the limitations of traditional diagnostic methods by leveraging diverse datasets to ensure comprehensive skin cancer classification.

- **Development Methodology**

The development of this system will follow a User-Centered Design methodology, tailored to general users without extensive collaboration with medical professionals. This approach prioritizes creating an intuitive platform that evaluates the likelihood of skin cancer through AI while minimizing complexity for non-expert users. The methodology involves the following steps:

- **Requirement Analysis:** Collecting input from potential general users to understand their needs and expectations.
- **Design and Prototyping:** Developing user-friendly interfaces for image uploading and result visualization.
- **Implementation:** Integrating pre-trained machine learning models for image analysis and cancer risk evaluation.
- **Testing and Validation:** Conducting thorough tests to ensure prediction accuracy and reliability.
- **Deployment and Maintenance:** Launching a widely accessible web-based platform and updating the system regularly to enhance its performance and usability.

The methodology ensures that the system remains focused on providing a seamless user experience while delivering accurate results. Collaboration with medical professionals is optional and limited to post-evaluation consultation, ensuring that the primary emphasis is on empowering general users with AI-driven assessment.

- **User Characteristics**

- **Participants**

- **Description:** Participants are the primary users of the system who will upload skin lesion images and analyze the results for medical or research purposes.

- **Characteristics:**

- **Role:** Typically doctors, dermatologists, or healthcare professionals. In some cases, it may include individual users (patients) who want a preliminary analysis.

- **Technical Knowledge:** Basic computer skills such as uploading files and interpreting simple analysis results. No advanced technical expertise is required.
- **Primary Tasks:**
  - Upload skin lesion images for analysis.
  - View and interpret analysis results.
  - Save or download reports for further evaluation.
- **Access:** Desktop interface.

### 2.2.2 Client Admin

- **Description:** Client Admins are responsible for managing the local client-side system, ensuring that local models are properly trained and updated, and overseeing client-side interactions.
- **Characteristics:**
  - **Role:** Client-side system administrators who manage local infrastructure and ensure smooth functioning of federated learning processes on client devices.
  - **Technical Knowledge:** Familiarity with machine learning workflows, client-side federated learning setups, and model management tools (e.g., Kafka ML-Flow), along with basic understanding of client-server communication.
  - **Primary Tasks:**
    - Monitor and manage client-side updates to the model.
    - Ensure that local client data is processed securely without compromising privacy.
    - Ensure smooth communication between client devices and the central server.
  - **Access:** Client UI with permissions to manage local models and configurations.

### 2.2.3 Server Admin

- **Description:** Server Admins are responsible for maintaining the central server, ensuring that federated learning models are properly updated, and managing server-side interactions and data aggregation from multiple client devices.

- **Characteristics:**
  - **Role:** System administrators who maintain the server-side infrastructure and ensure the federated learning system is running smoothly on the central server.
  - **Technical Knowledge:** In-depth knowledge of machine learning workflows, federated learning protocols, system security, and server maintenance. Familiar with server-side management tools such as Kafka-ML, and data aggregation techniques.
- **Primary Tasks:**
  - Monitor and manage the central server's functionality.
  - Aggregate updates from client devices and apply those updates to the global model.
  - Resolve issues related to client-server communication and system security.
  - Access: Server UI with advanced settings for managing federated learning processes and server-side configurations.

### 3. REQUIREMENTS SPECIFICATION

#### 3.1. External Interface Requirements

- **User interfaces**
  - The user interface will be designed to work on desktop and web-based platforms.
  - The system will have an intuitive and user-friendly design that will facilitate users to upload images, view analysis results, and access past analysis results.
  - The user interface will be developed with tools such as Tkinter or PyQt5.
- Hardware interfaces**
  - The system requires a standard USB port on the user's computer for uploading skin lesion images.
  - Hardware requirements include a minimum of 8 GB RAM and support for CPUs and GPUs capable of running machine learning models.

- An internet connection (Ethernet or Wi-Fi) is necessary for updating federated learning models.

- **Software interfaces**

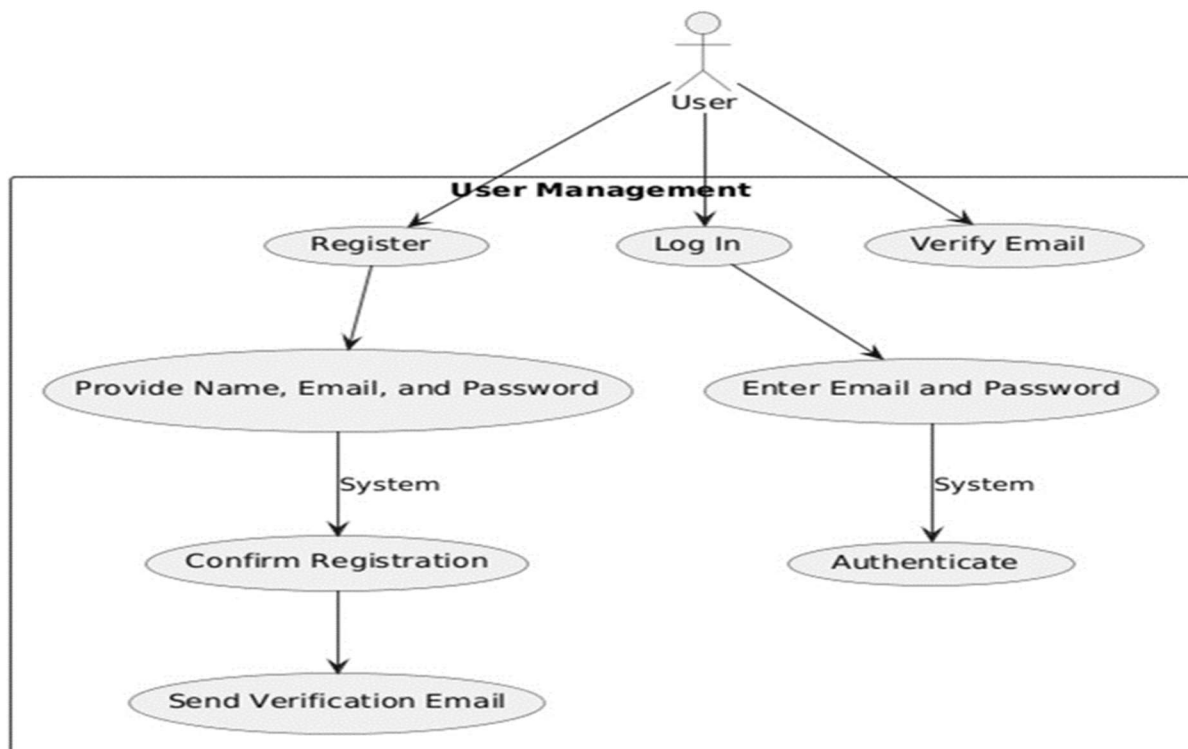
The system will be designed to run on major operating systems and should integrate with Python-based machine learning libraries. Secure protocols will be used to interact with external APIs, and the system will utilize federated learning for model training and updates without sharing raw data.

- **Communications interfaces**

The system will use HTTPS with TLS/SSL encryption for secure communication between the server and client devices. Communication will include sending model updates, receiving training status, and downloading the global model.

## 3.2 Functional Requirements

### 3.2.1. USER REGISTER USE CASE



#### Preconditions:

- The user must have access to the registration system.
- All required fields (name, email, password) must be entered correctly.



**Brief Description:** This use case allows new users to create an account by providing their name, email, and a password. The system validates the provided data, confirms the registration, and sends a verification email.

**Steps:**

- **User Action:** The user provides their name, email, and password in the registration form.
- **System Validation:**
  - The system checks the format and uniqueness of the email.
  - Ensures the password meets the complexity requirements (e.g., minimum length, character types).
- **Confirmation:**
  - The system confirms the registration and stores the user's data securely.
- **Email Verification:**
  - A verification email is sent to the user with a link or code to activate the account.
- **Activation:**
  - The user clicks the verification link or enters the provided code.
  - The system activates the user account, allowing login and access to additional features.

**Postconditions:**

- The user account is successfully created and activated.
- The system acknowledges the registration and redirects the user to the login page.

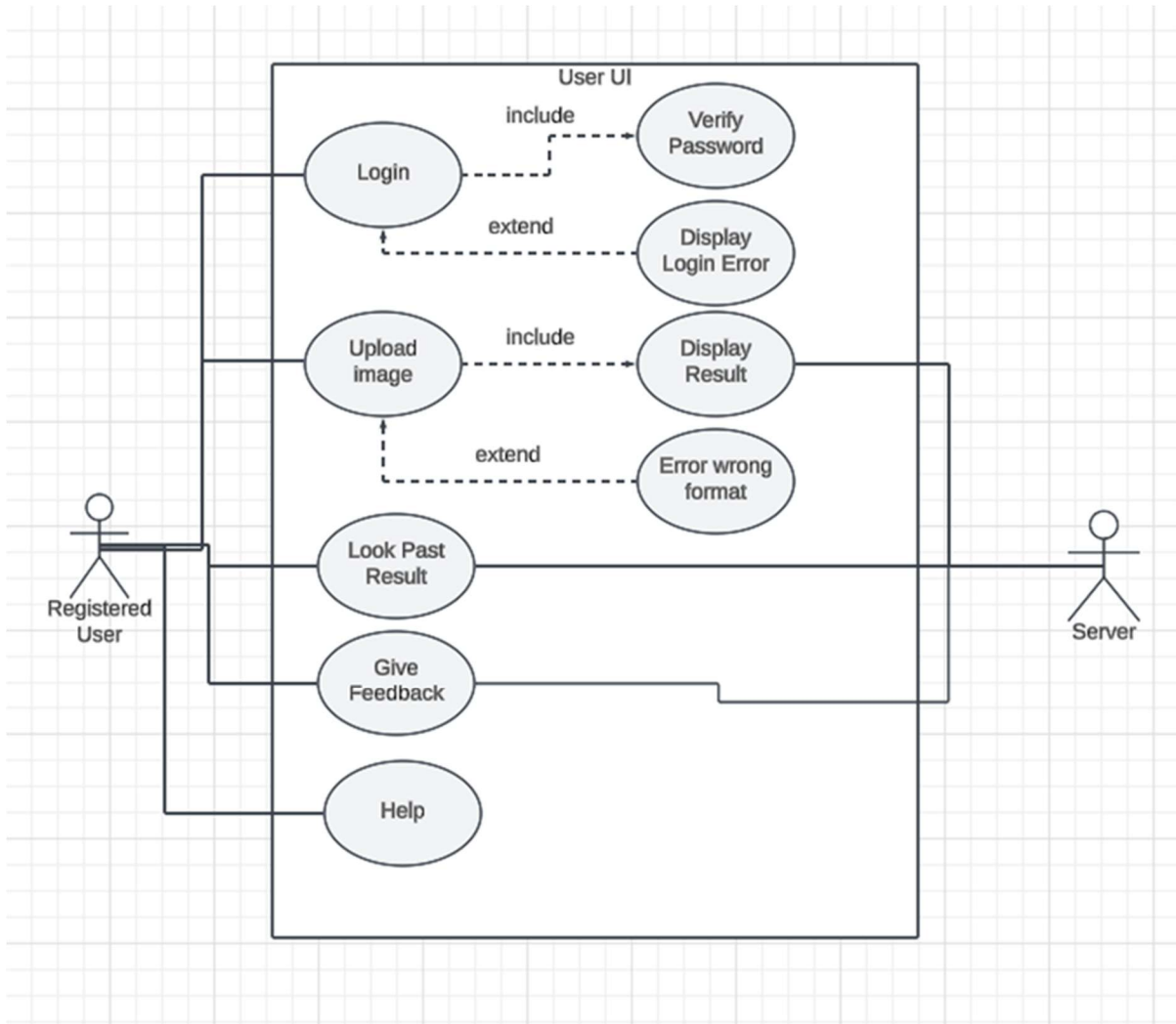
### 3.2.2 USER INTERACTION FLOW USE CASE

**User case:**

- Login
- Upload image
- Look Past Results

- Give Feedback
- Help

**Diagram:**

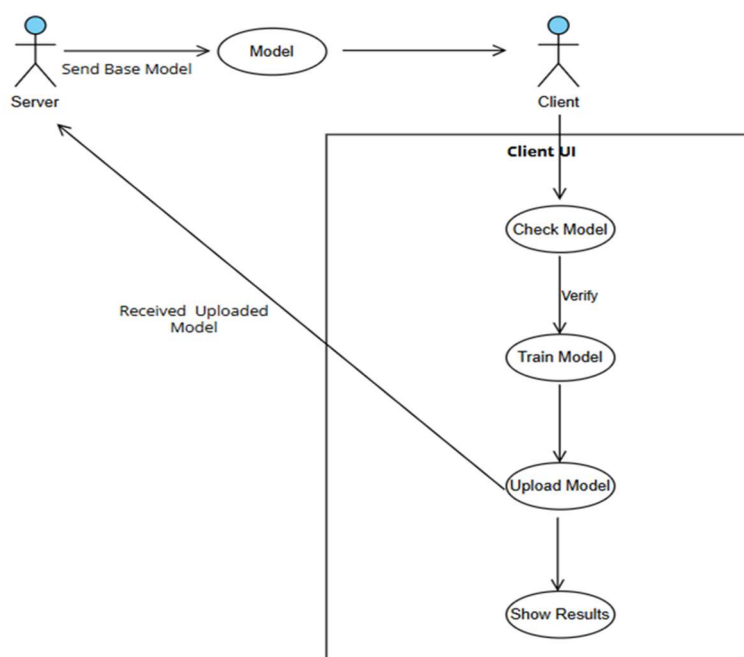


**Brief Description:** Figure 2 illustrates a Use Case diagram of the user's homepage screen. When the user accesses the homepage, they can take a photo of their skin and upload it to the system to check their likelihood of having skin cancer. The system analyzes the uploaded photo, calculates the cancer probability, and displays the result on the screen. Additionally, the user can view the results of their previous analyses in the "Past Results" section. Moreover, the user can provide feedback about the system to share their experience.

- If the login is successful, the user is redirected to the homepage.

- If the login fails, a message such as "Incorrect password" or "Incorrect email" is displayed on the screen.
- After accessing the homepage, the user is prompted to upload a photo of their skin and click the "Evaluate" button.
- If the uploaded photo is not in png or jpg format, an error message appears, indicating the incorrect file format.
- If the photo format is correct, the system proceeds to analyze the image. The user is informed that the analysis is in progress, and once completed, the results are displayed on the screen.
- Once the analysis is complete, the results are displayed on the screen.
- The user can also view their past results by clicking on the "Past Results" section on the homepage, which displays a list of previous evaluations. Each entry includes relevant details such as the date and analysis outcome.
- After completing all tasks, the user can leave feedback by navigating to the "Feedback" section, where they can provide their comments and suggestions.
- If the user needs assistance, they can click on the "Help" section for guidance, which provides instructions and answers to frequently asked questions.

### 3.2.3 CLIENT USE CASE



## Preconditions

Base Model Availability: The server must have a base model ready to send to the client.

- Client Connectivity: The client must be connected to the server to receive the base model.
- Data Readiness: Training data must be available on the client-side and properly formatted for model training.
- Client UI Accessibility: The client user interface must be operational for initiating and monitoring the process.
- Resource Availability: The local device should meet the computational requirements for model training (e.g., sufficient memory, CPU/GPU power).

**Brief Description:** This use case describes the process by which a client user trains a model locally using provided data. Once the model is trained, it is uploaded to a server. The results of the training process are displayed to the user for review.

## Steps:

☐Server Sends Model:

- When the system runs for the first time: The server sends the base model to the client device through the model component.
- On subsequent invocations: The server sends the last updated model to the client.
- The client user verifies that the model (base or updated) has been received.
- The client user initiates a check for the availability and validity of the model and training data through the client UI.
- The system ensures that the model and training data are present and valid for training.
- The client user starts the training process through the client UI. ( The system utilizes local resources to train the model using the provided data.)
- The trained model is sent back to the server.
- Show the results.

## Postconditions

☐Model Training Completion: The model is successfully trained using local data on the client device.

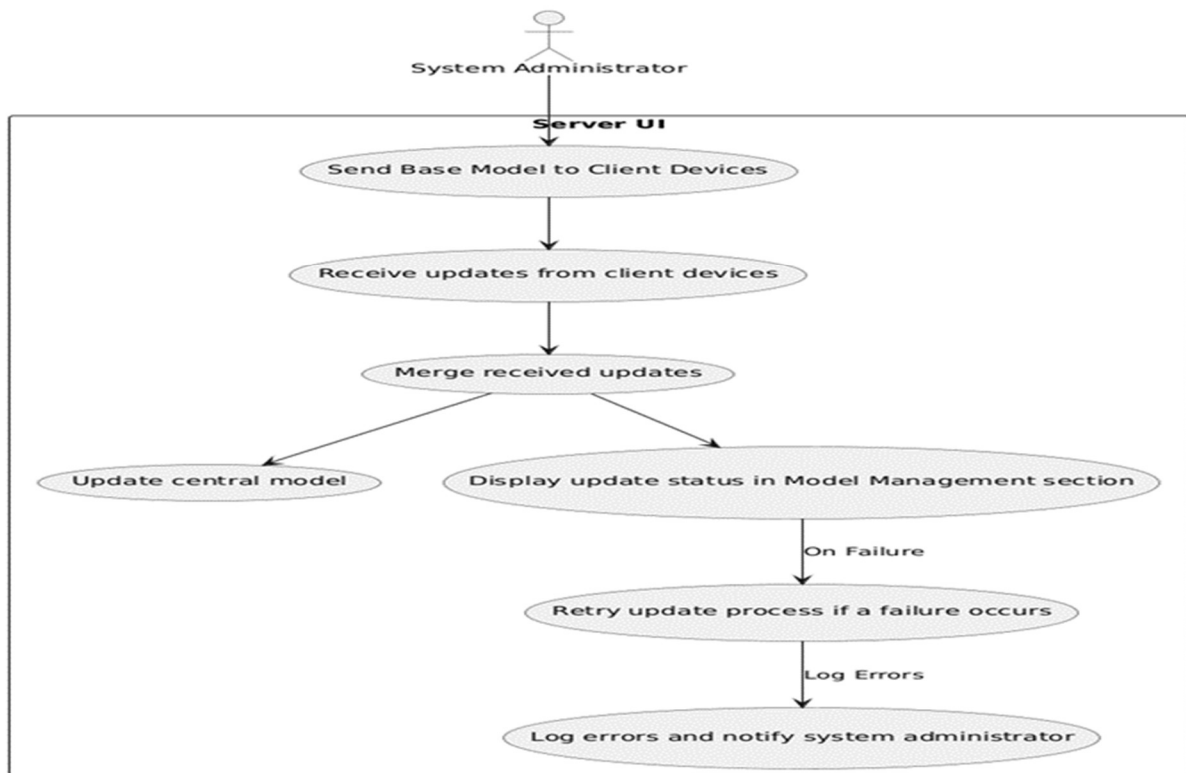
- Trained Model Uploaded: The trained model is sent to the server for further use, aggregation, or deployment.
- Training Results Displayed: The client user is able to view the results of the training process.

### 3.2.4 Server Management Use Case

#### Use Case:

- ☑ Send Base Model to Client Devices
- Receive updates from Client Devices
- Merge received updates
- Update central model
- Display update status in Model Management section
- Retry update process if a failure occurs
- Log errors and notify system administrator

#### Diagram:



## Preconditions

- Base Model Must Be Available:**

- The base model must be ready and accessible on the server before sending it to clients.

- Client Devices Must Be Connected:**

- All participating client devices must be connected to the server and actively transmitting updates.

- Central Server Must Be Running:**

- The central server must be operational and capable of receiving and processing updates.

- Valid Update Data:**

- The updates sent by the client devices must be in the correct format and validated for integrity (e.g., through hash checks).

## Brief Description of the Use Case

This use case represents the key operations performed within the Server UI in a federated learning system to manage and update the central model effectively. The primary actor is the System Administrator, who oversees the process. The use case involves the following steps:

- Send Base Model to Client Devices:** The Server UI sends the base model to the client devices. The base model includes initial weights and architecture, which the clients will use to train locally on their datasets.

- Receive Updates from Client Devices:** The server collects model updates (e.g., gradients or weights) sent by client devices participating in the federated learning process.

- Merge Received Updates:** The collected updates from multiple client devices are merged using a defined aggregation algorithm (e.g., weighted average) to create a unified update for the central model.

- Update Central Model:** The merged updates are applied to the central model to improve its performance and accuracy.

- Display Update Status:** The system displays the status of the update (e.g., success or failure) in the Model Management section of the Server UI for the administrator to review.

- Handle Failures:**

- If the update process encounters any failures, the system retries the process.
- In case of persistent failures, the system logs the errors and notifies the system administrator, ensuring transparency and providing insights for troubleshooting.

### 3.3 Performance Requirement

The system must meet the following performance requirements to ensure accurate and efficient skin cancer detection while providing a seamless user experience:

- Image Analysis Speed:** The analysis of a single image should be completed within a maximum of 10 seconds to ensure quick feedback for the users.
- Model Update Latency:** Model updates and weight transfers must occur with minimal delay, with a maximum latency of 5 seconds during federated learning processes.
- Model Accuracy:** The central model's performance must maintain an accuracy rate of at least 90% to ensure reliable results.

- Resource Optimization:**

- CPU and memory usage should be optimized during model training and image analysis to prevent system overloads.
- Libraries like TensorFlow [1], PyTorch[2], and NumPy will be used to enhance the computational efficiency of the system.

- Low-Latency Communication:**

- Tools like Kafka-ML [3] will be utilized to ensure low-latency data transfers between client and central servers in the federated learning process.

- Hardware Requirements:**

- Minimum 8 GB RAM to run the application without interruptions.
- Multi-core processors to support parallel processing during training and inference tasks.

## **3.4 Software System Attributes**

### **3.4.1. Portability**

- The application must run seamlessly on different platforms (Windows, macOS, Linux).
- The user interface must be compatible with both desktop and tablet devices.
- The federated learning system must support client devices with various operating systems (e.g., Windows, Linux).
- Tools like PyInstaller can be used for cross-platform distribution, and Docker containers can enhance the portability of the central server.

### **3.4.2. Performance**

- The analysis of a single image must be completed within a maximum of 10 seconds.
- Model updates and weight transfers must occur with minimal delay (maximum latency: 5 seconds).
- The central model's performance must maintain an accuracy rate of at least 90%.
- CPU and memory usage must be optimized during training.
- Libraries like NumPy, TensorFlow, or PyTorch can enhance system performance, and Kafka-ML can ensure low latency during data transfer.

### **3.4.3. Usability**

- The user interface must be intuitive and easy to understand.
- Analysis results must be presented clearly with visual support (e.g., charts, color-coded outcomes).
- Users must receive clear error messages for invalid inputs or uploads.
- A user-friendly help section or tutorial must be provided.
- Frameworks like Tkinter or PyQt5 can be used to create an accessible UI, and user testing can validate the design's intuitiveness.

### **3.4.4. Adaptability**

- The software must support the inclusion of new skin cancer types with minimal changes.
- New federated learning protocols or model architectures must be easy to integrate.



- The system must adapt to new data uploaded by users through incremental learning.
- A modular software architecture must be implemented to ensure flexibility, and incremental learning algorithms should be seamlessly integrated.

#### **3.4.5. Scalability**

- The system must be capable of handling updates from multiple client devices simultaneously, with scalability options to increase capacity as needed. Initially, the system should support at least 10-20 clients for testing and small-scale deployment, with plans to expand based on infrastructure upgrades and user demand.
- The central server must scale to accommodate models working with diverse datasets.
- Analysis processes must remain efficient even with large datasets.

#### **3.4.6. Security**

##### **•Data Privacy and Image Integrity:**

- The system must ensure that only valid skin lesion images are uploaded by users. If any irrelevant images (e.g., images of non-skin-related objects) are detected, the system must reject them and notify the user with an error message such as "Invalid image format. Please upload a valid skin lesion image."
- To prevent the misuse of the platform, all images should be checked for file integrity, size, and format (e.g., JPEG, PNG) before processing. The system must not accept any other file types or corrupted images.
- All user-uploaded images should be processed locally on the client device, ensuring no raw image data is transmitted to the central server. Only model updates (e.g., gradients) should be sent to the central server, preserving user privacy and compliance with data protection regulations (e.g., GDPR).

##### **•Security Measures for Data Transmission:**

- All data exchanged between the client devices and the central server must be encrypted using secure communication protocols (e.g., TLS/SSL). This ensures that the data (including model updates) cannot be intercepted or altered during transmission.
- The system must implement authentication and authorization protocols to ensure that only authorized clients can interact with the server. Each client device should authenticate with a secure token or similar mechanism before sending model updates to the central server.

##### **•Handling Irrelevant or Malicious Files:**

- If a user uploads an image that is flagged as irrelevant or malicious (e.g., containing harmful content), the system should immediately stop the analysis process and display a clear message to the user: "The image could not be processed due to security reasons."

- The system should log any such incidents to provide a record of attempted misuse for system administrators.

- Handling Timeouts and Long Processing:**

- If an image analysis takes longer than 10 seconds (e.g., due to system load or large image size), the system should automatically abort the process and prompt the user with a message such as, "The analysis could not be completed within the expected time. Please try again."

- Users should have the option to either retry the analysis or upload a smaller image.

- In case of system failure or excessive processing time, the system should not automatically restart. Instead, it should prompt the user with a notification that allows them to either retry or abort the analysis.

- A timeout mechanism should be implemented to ensure that the analysis process does not hang indefinitely, improving the user experience and preventing system overload.

- Malware and Threat Detection:**

- The system should include malware scanning of uploaded files to prevent the risk of virus or malware attacks via file uploads.

- Any suspicious activity or malicious attempts to upload harmful content should be logged and reviewed by the system administrator for further action.

- System Recovery in Case of Errors:**

- If the system encounters an unexpected error during image processing (e.g., timeout, memory overload, or corrupted data), it should gracefully handle the failure by providing a user-friendly error message.

- The system should include a recovery option, allowing users to retry the operation, but should never force a system restart or cause data loss.

### **3.5 Safety Requirement**

#### **Prevention of Misdiagnosis:**

- The system must clearly state that the analysis results are intended as a supportive tool and do not substitute professional medical diagnosis.

- Results should include a disclaimer such as: "This is not a diagnosis, but a probability analysis."

### **Secure Data Processing:**

- Images uploaded by users must be processed only on the local device and never sent to a central server.
- Data must not be shared with other users or systems under any circumstances.

### **Reliability of Analysis:**

- The system must provide the model's accuracy rate and confidence score with every analysis result.
- If the confidence score falls below a certain threshold (e.g., 90%), the user should be notified with a warning.

### **Protection Against User Errors:**

- Unsupported file types (e.g., PDFs or text files) or invalid inputs must trigger error messages to guide the user.
- Before starting an analysis, the system should inform the user about acceptable file formats and requirements.

### **Backup and Recovery During System Failures:**

- If the system crashes or encounters an interruption during analysis, the process must be automatically saved, allowing the user to resume from where they left off.
- Local training processes should also be restartable in case of interruptions.

### **Critical System Performance:**

- The system must be optimized to avoid overheating or crashes under heavy workloads (e.g., analyzing large datasets).
- Performance monitoring tools should regularly check the system's status.

### **Secure Updates:**

- Model updates from the central server must be verified for integrity (e.g., through hash checks) before being applied.

- A backup version of the model must always be available to mitigate risks associated with faulty updates.

**Protection Against Misuse:**

- The system must validate the type and format of uploaded data to prevent users from uploading irrelevant or inappropriate files.
- For instance, only image files (JPEG, PNG) should be accepted.

# Software Design Document

## 1. Introduction

### 1.1 Purpose

This document serves as a comprehensive guide to the design and architecture of the Skinalyzer: AI-Based Skin Cancer Detection System. It aims to detail the system's core components, structural framework, and workflow processes, providing a blueprint for its seamless implementation. By integrating advanced techniques such as Federated Learning and Incremental Learning, the system achieves a perfect balance of privacy, adaptability, and high diagnostic precision.

### 1.2 Scope

The scope of this project includes the development of a desktop application for diagnosing skin cancer using machine learning. The system will analyze skin lesion images uploaded by users, ensuring privacy through local data processing. In addition to uploading images from desktop devices, users will also have the ability to upload photos directly from their smartphones, enhancing accessibility and convenience. A centralized server will aggregate updates from client devices to refine a global model, while Incremental Learning ensures continuous improvement with new data. Designed for healthcare professionals and individual users, the system prioritizes usability, scalability, and secure communication.

### 1.3 Glossary

**AI-Based Skin Cancer Detection System:** A system utilizing Artificial Intelligence (AI) to analyze skin lesion images for early detection of skin cancer.

**Federated Learning (FL):** A machine learning approach where data remains on client devices, and only model updates (e.g., gradients) are shared with a central server to maintain privacy.

**Incremental Learning (IL):** A technique that allows machine learning models to learn new data while retaining knowledge from previously trained data.

**Central Server:** A server that aggregates model updates from client devices, refines a global model, and redistributes it to clients.

**Client Device:** A user's local device (e.g., desktop, mobile) used for uploading skin lesion images, processing data, and training models locally.

**Global Model:** A centralized machine learning model that combines updates from multiple clients to improve performance and accuracy.

**Kafka-ML:** A message broker used to manage real-time communication and data exchange between client devices and the central server.

**MLFlow:** A tool for tracking and managing machine learning models, including versioning and performance metrics.

**Analysis Result:** The outcome of processing and analyzing a skin lesion image, including the diagnosis and confidence score.

**Skin Lesion:** A visible abnormality on the skin, which may indicate the presence of skin cancer.

**Gradient Data:** The data generated during the training of a machine learning model, used for updating the global model in federated learning.

**Login History:** Records of users' login attempts, including timestamps and user identifiers.

**Authentication:** The process of verifying a user's identity using credentials like email and password.

**Verification Email:** An email sent to users during the registration process to confirm their email address.

**Base Model:** An initial machine learning model distributed to clients for local training in a federated learning workflow.

**Data Repository:** A centralized storage system for user-uploaded images and other data needed for future analysis or training.

**Update Log:** A log of changes and updates made to the global model, including contributions from specific client devices.

**User Interface (UI):** The visual component of the system that allows users to interact with features such as image uploads and result viewing.

## 1.4 Overview of Document

This document provides a structured and detailed outline of the design specifications for the Skinalyzer: AI-Based Skin Cancer Detection System. It begins with an introduction to the project, highlighting its purpose, scope, and objectives. The document then delves into the system's architecture, providing a high-level overview of its components, including the user interface, client-side modules, server-side modules, and the database.

One of the key aspects of this document is the detailed design of the user interface (UI). The UI has been carefully crafted using Figma, a collaborative interface design tool, to ensure a user-friendly and intuitive experience. The design process focuses on enabling seamless navigation for users, incorporating essential functionalities such as uploading images, viewing analysis results, accessing past diagnoses, and providing feedback. Visual prototypes created in Figma are included to offer a clear representation of the intended look and feel of the application.

Subsequent sections detail the design methodologies, workflows, and communication mechanisms that ensure the system operates efficiently and securely. Diagrams such as the architectural layout, class diagrams, and entity-relationship diagrams (ERDs) are included to offer visual clarity and a deeper understanding of the system's structure.

The document also covers the integration of key machine learning technologies, including Federated Learning and Incremental Learning, and how these are leveraged to ensure privacy, scalability, and continuous improvement of the model. Finally, it provides implementation details for user interactions, model training, and system maintenance, serving as a practical guide for developers and stakeholders throughout the project's lifecycle.

## 1.5 Motivation

The Skinalyzer: AI-Based Skin Cancer Detection System is motivated by the critical need for early and accurate diagnosis of skin cancer, which can significantly improve patient outcomes and survival rates. Skin cancer remains one of the most common types of cancer worldwide, and its early detection is often hindered by the lack of accessible, reliable, and affordable diagnostic tools. This project aims to bridge this gap by leveraging cutting-edge machine learning techniques to provide an efficient, user-friendly, and privacy-preserving diagnostic solution.

The integration of Federated Learning ensures that users' sensitive data, such as skin lesion images, remain securely on their local devices, aligning with modern privacy standards and increasing user trust. At the same time, Incremental Learning enables the system to adapt and evolve by learning from new data without losing prior knowledge, ensuring its effectiveness over time.

By designing a system that is accessible to both healthcare professionals and individual users, the Skinalyzer project aspires to democratize advanced diagnostic technologies. It seeks to empower users with actionable insights while reducing the burden on medical professionals, ultimately contributing to improved public health outcomes. The motivation behind this project

is rooted in making AI-driven healthcare innovations more inclusive, secure, and reliable for widespread use.

## 2. SYSTEM OVERVIEW

This project aims to develop a desktop application for supporting the diagnosis of skin cancer by analyzing skin lesion images. The system utilizes advanced machine learning techniques, including Federated Learning (FL) and Incremental Learning (IL), to ensure privacy, adaptability, and efficiency in model training and predictions.

The application provides a user-friendly interface where users can upload images, view analysis results, and access past diagnoses. The system processes uploaded images locally on client devices, contributing to the federated learning process, which ensures that raw user data remains on their devices, maintaining privacy and compliance with data protection standards.

The centralized server collects and aggregates updates from clients to improve the global model. Incremental learning allows the system to update the model with new data while preserving knowledge from previous datasets, ensuring continuous model enhancement without data loss.

The project integrates key components such as:

- **User Interface (UI):** A responsive and intuitive UI for users to interact with the system.
- **Client Module:** Handles local model training and communication with the server.
- **Server Module:** Manages global model updates and provides coordination for the federated learning workflow.

### 3.1 High-Level Architecture

The system operates in a decentralized manner, where client devices locally process and train models using user-provided data (skin lesion images). Updates to the model are communicated securely to a centralized server using Kafka-ML. The server aggregates these updates, improves the global model, and redistributes the updated model to the clients.

#### Key Components:

- **Clients:** Handle user interactions, local data processing, and model training.
- **Server:** Coordinates model aggregation, versioning, and distribution.
- **Kafka-ML:** Manages high-throughput, real-time communication of model updates between clients and the server.
- **MLFlow:** Tracks model versions, performance metrics, and manages the lifecycle of both local and global models.



## 3.2 Federated Learning Workflow

The Federated Learning workflow is at the core of the system. It ensures data privacy by keeping user data local while still enabling collaborative model improvement.

### 1. Model Initialization:

- o The server initializes a base model and distributes it to all clients using Kafka-ML.

### 2. Local Training:

- o Clients train the model locally on their datasets (skin lesion images).
- o Incremental Learning ensures that new training data is incorporated without losing knowledge from previous datasets.

### 3. Model Update Communication:

- o Clients send model updates (gradients) to the server using Kafka-ML.

### 4. Global Model Aggregation:

- o The server aggregates updates using Federated Averaging or similar techniques.
- o The aggregated model is stored and versioned in MLFlow.

### 5. Redistribution:

- o The updated global model is sent back to clients for further training cycles.

## 3.3 Architectural Diagram

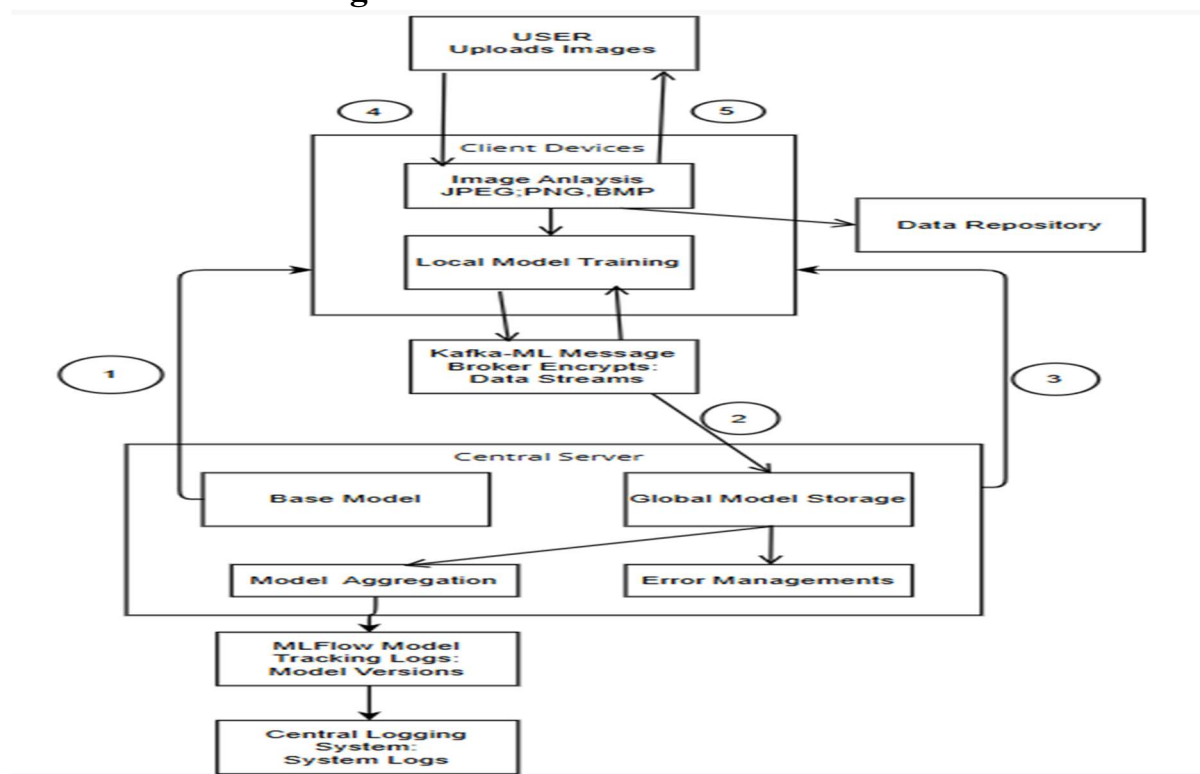


Figure 1: Architectural Diagram

1. We send the base model to multiple clients and train the model.
2. We combine the trained models to create a global model.
3. We send the global model back to the clients.
4. The user uploads a photo for testing

**Loop:** Is the Data Repository empty?

- **Yes:**
  - Step 4: The photos accumulate in the repository.
  - Step 5: The client analyzes the photo using the model and shows the result to the user.
- **No:**
  - Go back to Step 2.

## 1. User

- **Action:** The user uploads skin lesion images to the system through the user interface.
- **Purpose:** Users can view the analysis results and upload new images for diagnosis.

## 2. Client Devices

- **Components:**
  - **Image Analysis:**
    - Processes uploaded images.
    - Supported formats include JPEG, PNG, and BMP.
    - Ensures compatibility with a wide range of image types.
  - **Local Model Training:**
    - Performs training of the model using local data on the client devices.
    - Sends model weights or gradient updates to the server for aggregation.
- **Role:** Acts as the intermediary between the user and the central server, handling local data processing and model updates.

## 3. Kafka-ML Message Broker

- **Function:**
  - Encrypts data streams to ensure secure communication.
  - Facilitates the transfer of the base model from the central server to the client devices.
  - Handles updates sent back from clients, ensuring reliable and efficient data transfer.

**Purpose:** Serves as the communication hub, enabling decentralized model updates and data exchange.

#### 4. Data Repository

- **Function:**

- Stores processed images for future reference or analysis.
- Provides a centralized location for storing user-uploaded data.

**Role:** Supports data persistence and allows historical analysis if needed.

#### 5. Central Server

**Components:**

- **Model Aggregation:**
  - Aggregates model updates (gradient data) from multiple client devices.
  - Combines the updates to improve the global model.
- **Error Management:**
  - Identifies and handles faulty updates or issues with client-server communication.
  - Ensures stability and reliability of the aggregation process.

**Role:** Acts as the core of the federated learning system, orchestrating updates and maintaining the global model.

#### 6. Global Model Storage

- **Function:**

- Stores the aggregated global model.
- Updates the global model when a predefined threshold is reached (e.g., after receiving enough updates from clients).

**Purpose:** Serves as the main storage for the machine learning model used across all devices.

#### 7. MLFlow Model Tracking

**Function:**

- Logs versions of the global model for tracking and reproducibility.
- Monitors the model's evolution over time.

**Purpose:** Supports experiment tracking, making it easier to analyze the performance and development of the model.

#### 8. Central Logging System

- **Function:**

- Stores system logs, including events, errors, and updates.

- **Role:** Ensures system transparency and aids in debugging or auditing.

## 4. Detailed Design

### 4.1 Class Diagram

#### Class Descriptions:

1. **User**: Represents the user entity with attributes for name, email, and password. It includes methods for registration, login, profile update, image upload, and viewing analysis results.
2. **Image**: Represents images uploaded by users, with a method to analyze the image and update its status.
3. **AnalysisResult**: Represents the results of image analysis, including generating a report.

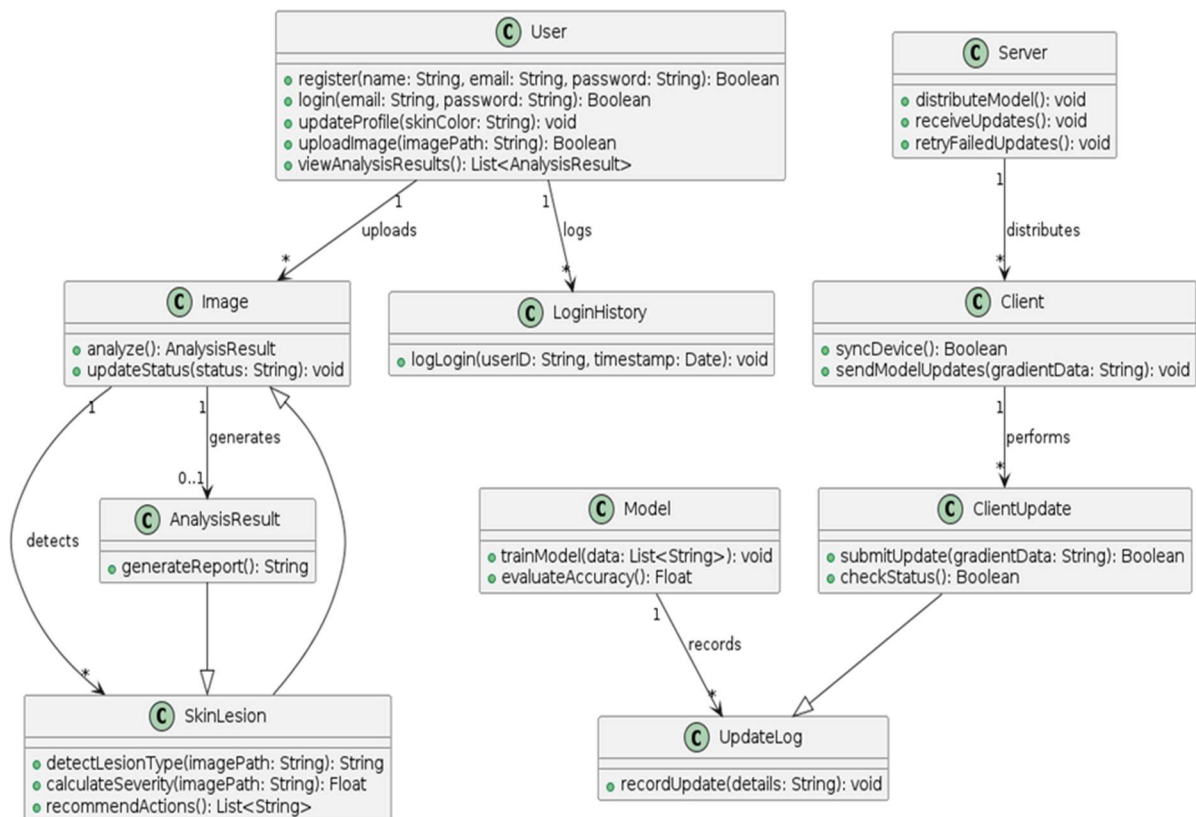


Figure 2: Class Diagram

1. **SkinLesion**: Represents skin lesions detected in images, with methods to detect lesion type, calculate severity, and recommend actions.

2. **LoginHistory:** Logs user login activities with user ID and timestamp.
3. **Model:** Represents the machine learning model used for training on data and evaluating accuracy.
4. **UpdateLog:** Logs updates for models, detailing the update specifics.
5. **ClientUpdate:** Represents updates sent from clients, including gradient data for federated learning.
6. **Client:** Represents client devices that sync with the server and send model updates.
7. **Server:** Manages the distribution of models, receiving updates, and handling failed updates in the federated learning system.

### Relationships:

- Users upload images, which are then analyzed, generating results that can detect skin lesions.
- Users' login activities are recorded in the login history.
- The model is trained with data and its accuracy is evaluated, with updates being logged.
- Clients perform updates which are sent to the server, and the server distributes these updates back to clients.

This class diagram captures the flow and interaction between different components in a federated learning system tailored for skin cancer analysis.

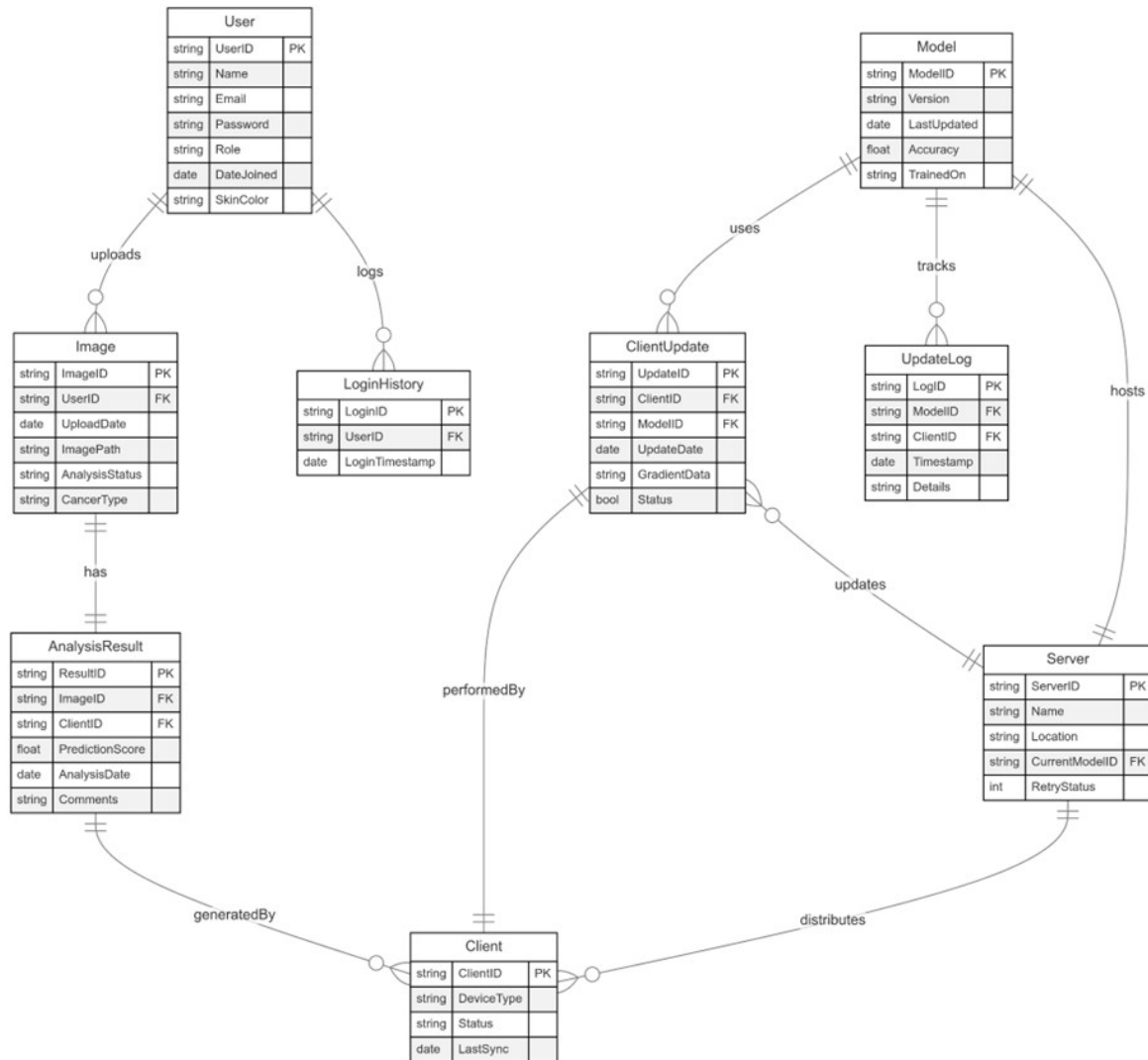
## 4.2 Entity Relationship Diagram (ERD)

This diagram is an **Entity-Relationship (ER) Diagram** representing the key entities, their attributes, and relationships in a system designed for federated learning and skin cancer analysis.

### Key Entities and Their Attributes:

#### 1. User

- **Description:** Represents the primary users of the system who upload images and view analysis results.
- **Attributes:**
  - UserID: Uniquely identifies each user.
  - Name, Email, Password, Role: Stores user credentials and roles.
  - DateJoined: Tracks when the user joined the system.
  - SkinColor: Captures skin color information, useful for specific analyses.



**Figure 3: Entity Relationship (ER) Diagram**

## 2. Relationships:

- **uploads**: A user uploads images stored in the Image table.
- **logs**: A user's login history is recorded in the LoginHistory table.

## 2. Image

- **Description**: Represents images uploaded by users for analysis.
- **Attributes**:
  - **ImageID**: Uniquely identifies each image.
  - **UserID**: References the user who uploaded the image.
  - **UploadDate, ImagePath**: Tracks the upload date and file path of the image.
  - **AnalysisStatus, CancerType**: Indicates the analysis status and detected cancer type.
- **Relationships**:
  - **has**: Each image is associated with an analysis result in the AnalysisResult table.

### 3. AnalysisResult

- **Description:** Captures the results of analyzing uploaded images.
- **Attributes:**
  - ResultID: Uniquely identifies each analysis result.
  - ImageID: Links the result to the analyzed image.
  - PredictionScore: Confidence level or probability of the prediction.
  - AnalysisDate: Date of the analysis.
  - Comments: Additional notes or remarks about the analysis.
- **Relationships:**
  - **generatedBy:** The analysis is performed by a Client device.

### 4. Client

- **Description:** Represents devices (e.g., desktops, mobile devices) used for local model training and analysis.
- **Attributes:**
  - ClientID: Uniquely identifies each client device.
  - DeviceType, Status, LastSync: Captures device type, operational status, and the last sync time.
- **Relationships:**
  - **performedBy:** A client performs the analysis for an image.
  - **generatedBy:** Links a client to the analysis it generates.

### 5. LoginHistory

- **Description:** Tracks the login history of users.
- **Attributes:**
  - LoginID: Uniquely identifies each login event.
  - UserID: References the user who logged in.
  - LoginTimestamp: Records the login time.
- **Relationships:**
  - **Logs:** Tracks which user performed the login.

### 6. Model

- **Description:** Represents machine learning models hosted on the central server.
- **Attributes:**
  - ModelID: Uniquely identifies each model.
  - Version, LastUpdated: Stores the version and last update timestamp of the model.
  - Accuracy, TrainedOn: Indicates model accuracy and training data source.
- **Relationships:**
  - **tracks:** Model updates are recorded in the UpdateLog table.

- **uses:** Model updates are linked to ClientUpdate.

## 7. ClientUpdate

- **Description:** Tracks updates sent by client devices to the central server.
- **Attributes:**
  - UpdateID: Uniquely identifies each update.
  - ClientID: Links the update to a specific client.
  - ModelID: Indicates the model being updated.
  - UpdateDate, GradientData: Tracks the update timestamp and data gradients sent.
  - Status: Indicates whether the update was successful.
- **Relationships:**
  - updates:** Links the updates back to the central server.

## 8. UpdateLog

- **Description:** Logs updates and changes made to machine learning models.
- **Attributes:**
  - LogID: Uniquely identifies each log entry.
  - ModelID: Links the log entry to a specific model.
  - ClientID: Tracks which client contributed to the update.
  - Timestamp, Details: Logs the date and details of the update.
- **Relationships:**
  - **tracks:** Maintains a history of all updates to the model.

## 9. Server

- **Description:** Represents the central server that aggregates updates and distributes the base model.
- **Attributes:**
  - ServerID: Uniquely identifies the server.
  - Name, Location: Metadata about the server.
  - CurrentModelID: Links the server to the currently active model.
  - RetryStatus: Tracks whether retry mechanisms are triggered for updates.
- **Relationships:**
  - **distributes:** The server distributes the base model to client devices.
  - **hosts:** The server hosts and aggregates updates from clients.

## Summary of Relationships:

- **Users** upload images, provide feedback, and are tracked via their login history.



- **Images** are analyzed, with analysis results linked to specific images and the clients that performed the analysis.
- **Login history** records user activity, tracking timestamps and user IDs.
- **Feedback** captures user input and ratings regarding the analysis results.
- **Client devices** perform local model training, generate analysis results, and send updates to the central server.
- **Client updates** contribute gradient data and model updates to improve the global model managed by the central server.
- The **server** aggregates model updates from clients, distributes base models to devices, and tracks model versions using logs.
- The **UpdateLog** records all updates to models, associating them with specific clients and maintaining a history for traceability.

## 5. Use Case Realization

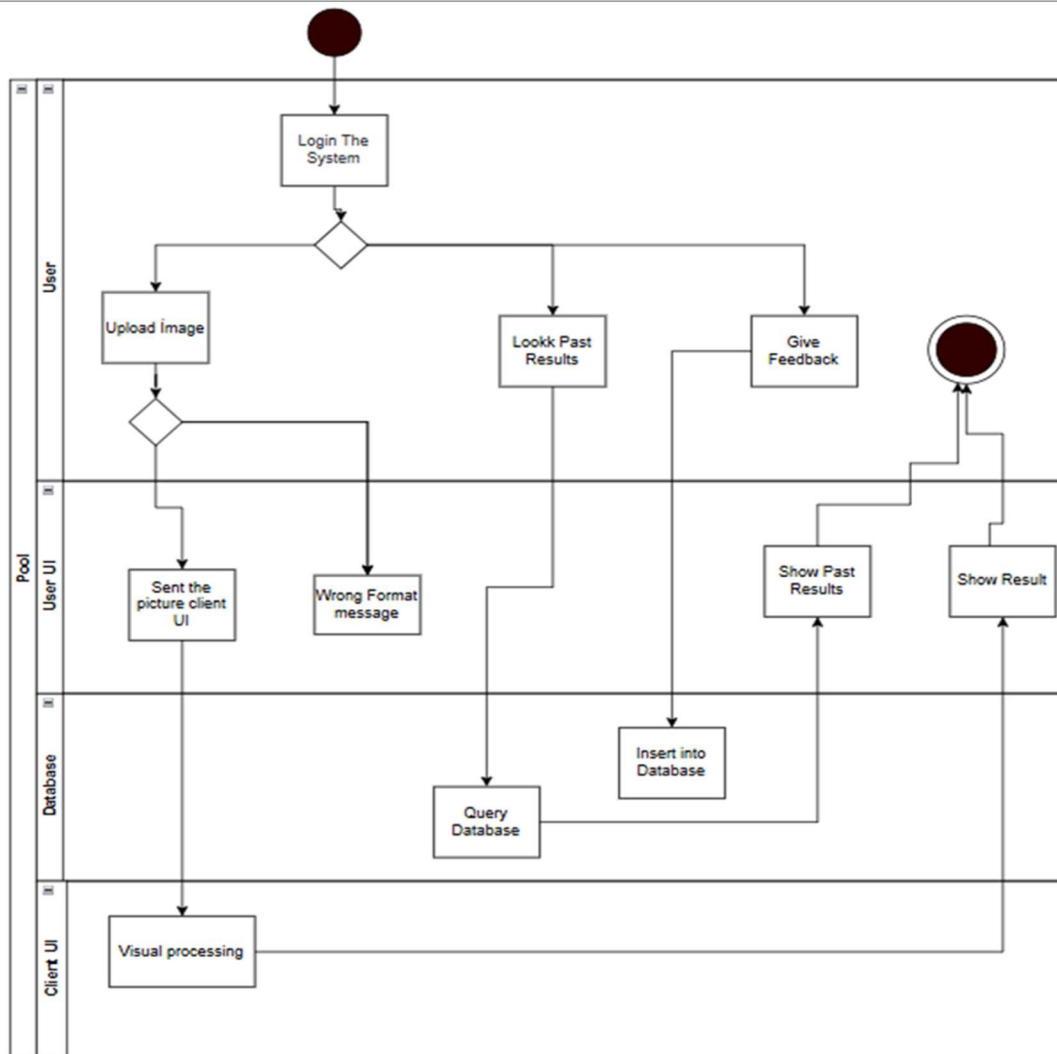
### 5.1 USER REGISTER FLOW



This class diagram represents the user registration and authentication process within a system. The process begins with the user providing necessary details, such as their name, email, and password, during registration. Once these details are submitted, the system confirms the registration and sends a verification email to ensure the provided email address is valid. The user must verify their email through the link or code provided in the email. After successful verification, the user can log in by entering their email and password. The system then authenticates the user's credentials to ensure they match the records. As an additional security measure, the system verifies the email before granting full access to the user. This sequential process ensures both user convenience and system security by validating user information at multiple stages.

## 5.2 USER INTERACTION FLOW

### Activity Diagram

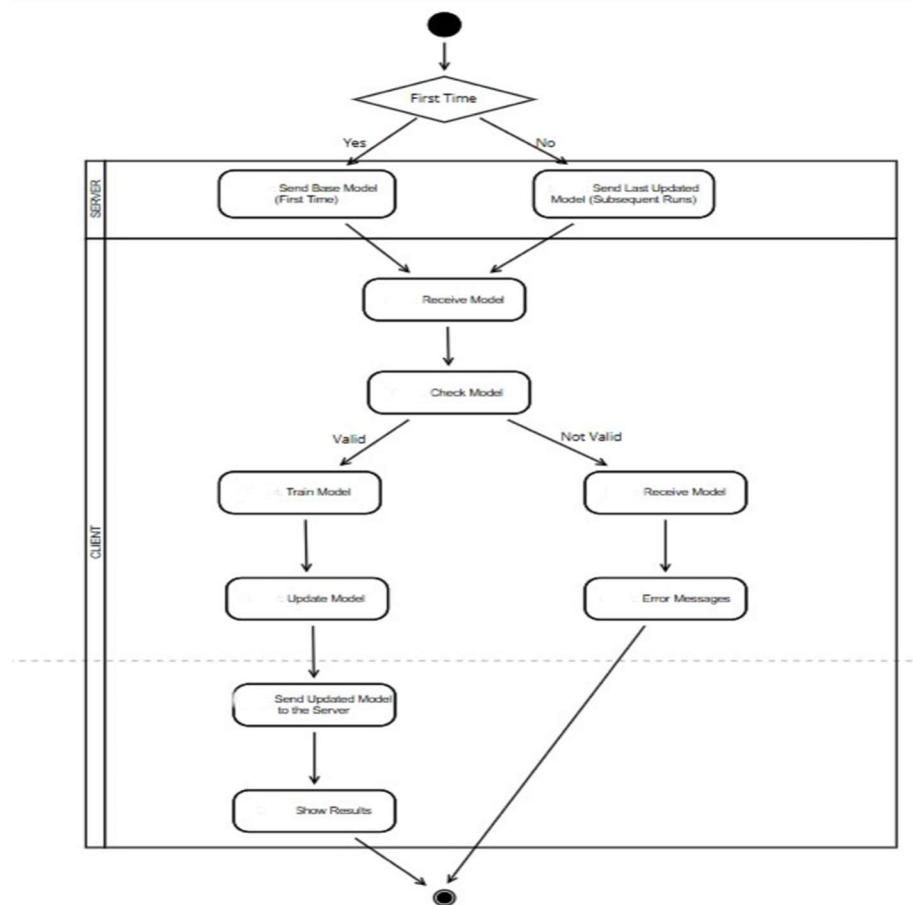


The activity diagram illustrates the workflow of the user's interaction with the system on the homepage. It begins with the user accessing the homepage after a successful login. The user is prompted to upload a photo of their skin, which triggers a validation process. If the file format is incorrect, an error message is displayed; otherwise, the system proceeds with the image analysis. While the analysis is ongoing, a progress notification is shown. Once completed, the results are displayed on the screen.

The user can then navigate to the "Past Results" section to view a list of previous evaluations, including dates and outcomes. Additionally, the user can provide feedback by accessing the "Feedback" section to share comments or suggestions. If assistance is needed, the "Help"

section offers guidance and FAQs. The activity diagram captures this sequence of actions and decision points, representing the system's dynamic behavior in response to user interactions.

### 5.3 CLIENT



### Overview of the Workflow

#### Start Node:

The process begins with the system being invoked.

#### Decision - First Time?:

A decision node determines if it is the first invocation of the system:

1. Yes → The server sends the Base Model.
2. No → The server sends the Last Updated Model from previous runs.

#### Receive Model (Client):

The client receives the model (Base or Updated) and proceeds with the process.

#### Check Model (Client):

The client checks the validity of the received model and training data.

#### Decision - Model Valid?

A second decision node evaluates if the model is Valid:

- Valid: The client proceeds to train and update the model.
- Not Valid: The client handles errors and may display appropriate error **messages**.

#### Valid Path:

- Train Model: The client trains the model using local data.

- Update Model: The client updates the model with the newly trained data.
- Send Updated Model to the Server: The client uploads the updated model back to the server.
- Show Results: The system displays the training results to the client user.

**Not Valid Path:**

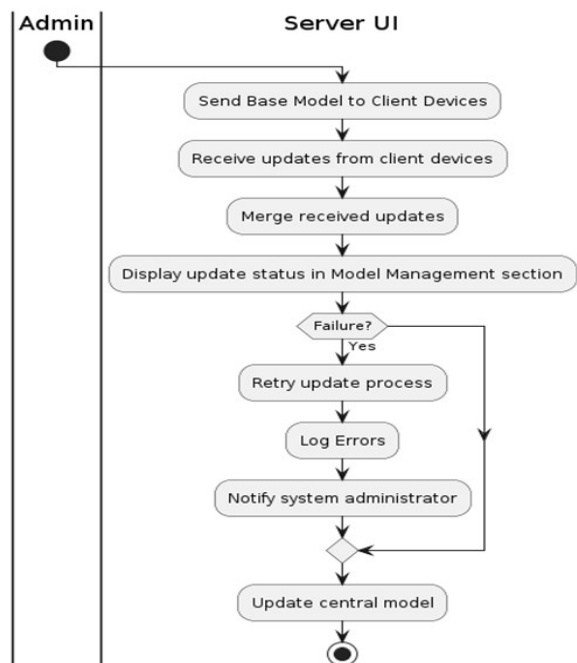
- Receive Model: The client re-attempts to receive the model (optional error-handling loop).
- Error Messages: The system displays error messages to inform the client of issues.

**End Node:**

The workflow concludes.

## 5.4 SERVER MANAGEMENT

### Activity Diagram:



### Flow of Operations

**1. Start**

- The process begins at the **Start Node**.

**2. Send Base Model to Client Devices**

- The system sends the base model to the connected client devices.

**3. Receive Updates from Client Devices**

- The system waits to receive updates from the client devices.

**4. Merge Received Updates**

- All received updates are merged into a central system.

### Parallel Execution Using Split

At this point, the flow splits into **two parallel activities**:

## 1. Update Central Model

- The merged updates are applied to the central model.

## 2. Display Update Status in Model Management Section

- The status of the updates is displayed in the **Model Management** section.
- A conditional check is performed:
  - **Failure Check:** If an error or failure occurs:
    - **Retry Update Process:** The system retries the update process.
    - **Log Errors:** Errors are logged for future analysis.
    - **Notify System Administrator:** A notification is sent to the system administrator.

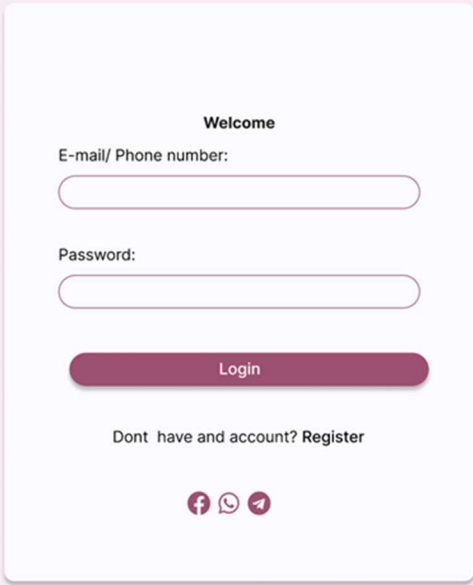
## End of Process

- After both parallel activities are completed, the flow merges back.
- The **Stop Node** indicates the end of the process.

# 6. User Interface Design

## 6.1 Login Section:

The user enters their email or phone number and password, then clicks the "Login" button. The backend validates the credentials against the database. If correct, the user is redirected to the homepage; otherwise, an error message appears.






Welcome

E-mail/ Phone number:

Password:

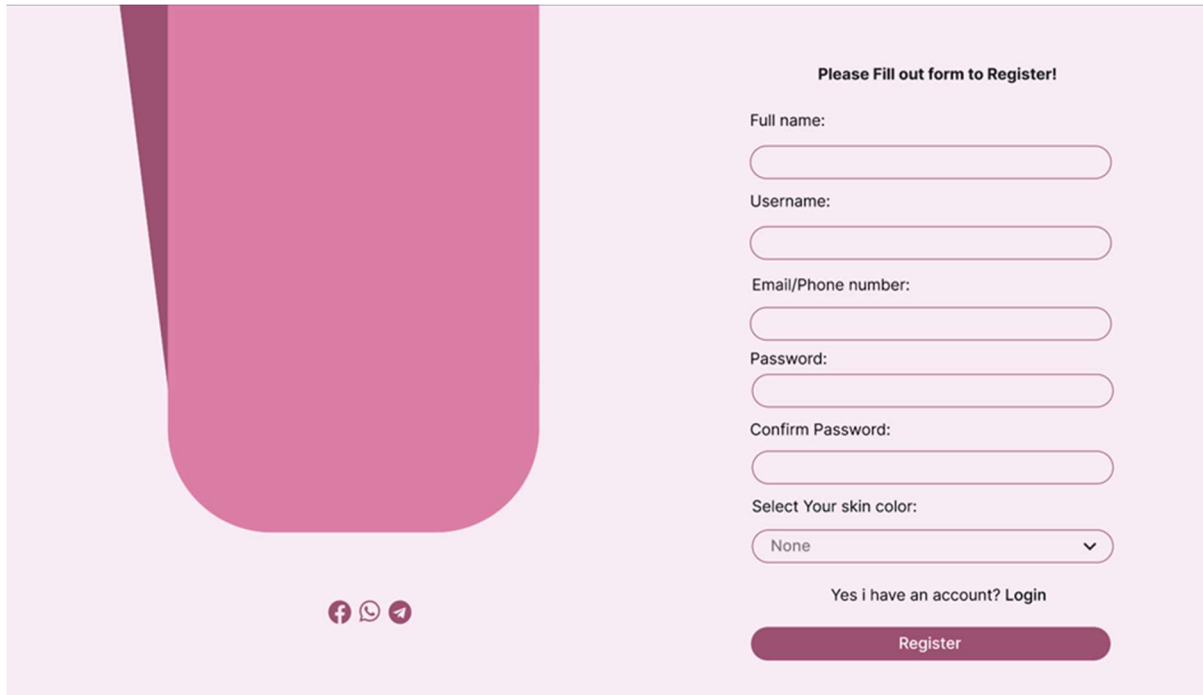
Login

Dont have and account? Register

## 6.2 Register Section:

The user provides their first name, last name, password, and skin tone, selecting the latter for skin cancer awareness. Upon clicking "Register," the backend checks if the email/phone number is already registered and verifies the password match. If valid, the user is added to the database and redirected to log in; otherwise, an error message appears.



The registration form is titled "Please Fill out form to Register!". It includes input fields for Full name, Username, Email/Phone number, Password, and Confirm Password. There is a dropdown menu for "Select Your skin color:" with "None" selected. A "Login" link is provided for users who already have an account. A "Register" button is at the bottom. Social media icons for Facebook, WhatsApp, and Telegram are located at the bottom left of the form area.

Please Fill out form to Register!

Full name:

Username:

Email/Phone number:

Password:

Confirm Password:

Select Your skin color:

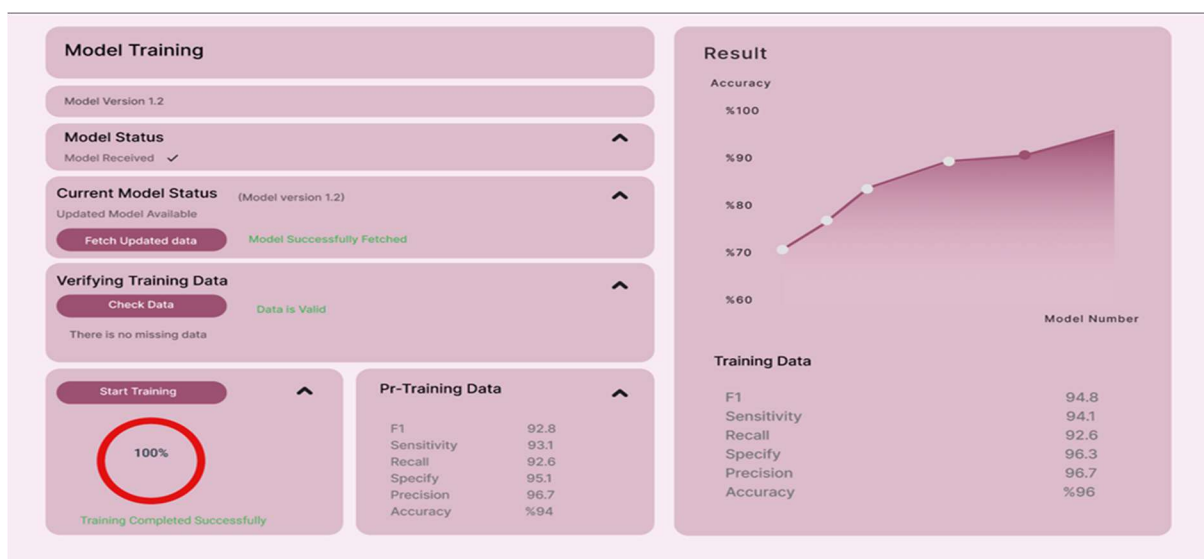
None

Yes i have an account? [Login](#)

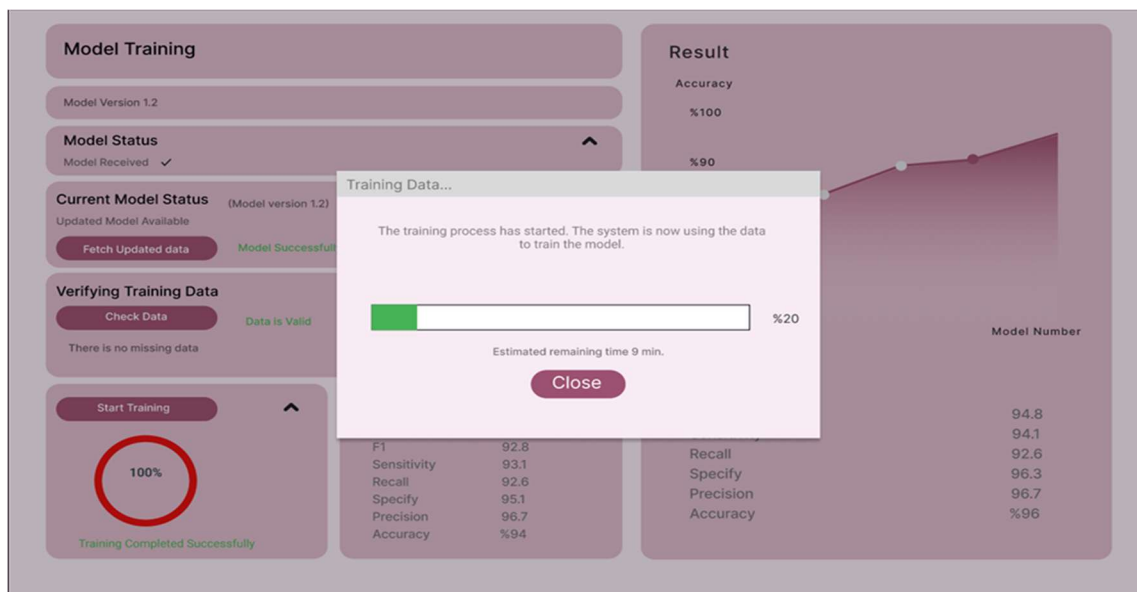
[Register](#)

## 6.3 Client UI:

The model is initially trained with available data and updated with new data over time, retaining previous knowledge. Updates use only new data to ensure efficient memory and resource management. This approach enables continuous learning by combining new information with existing knowledge

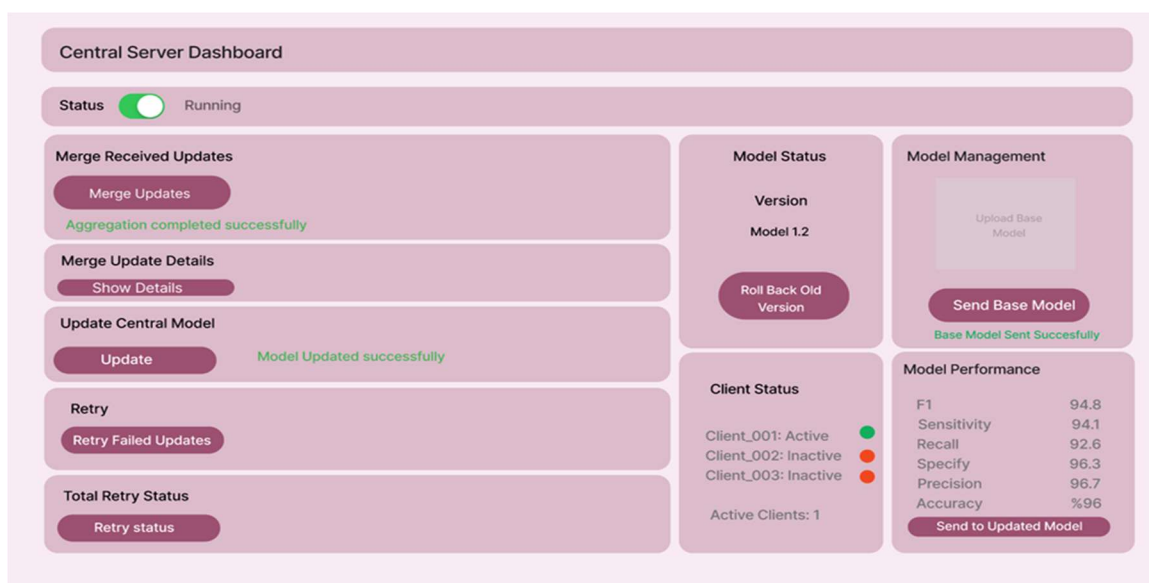


Users can fetch the latest dataset, verify its validity and structure, and start the training process locally after validation. Once training is complete, key metrics like accuracy and loss are displayed for review.

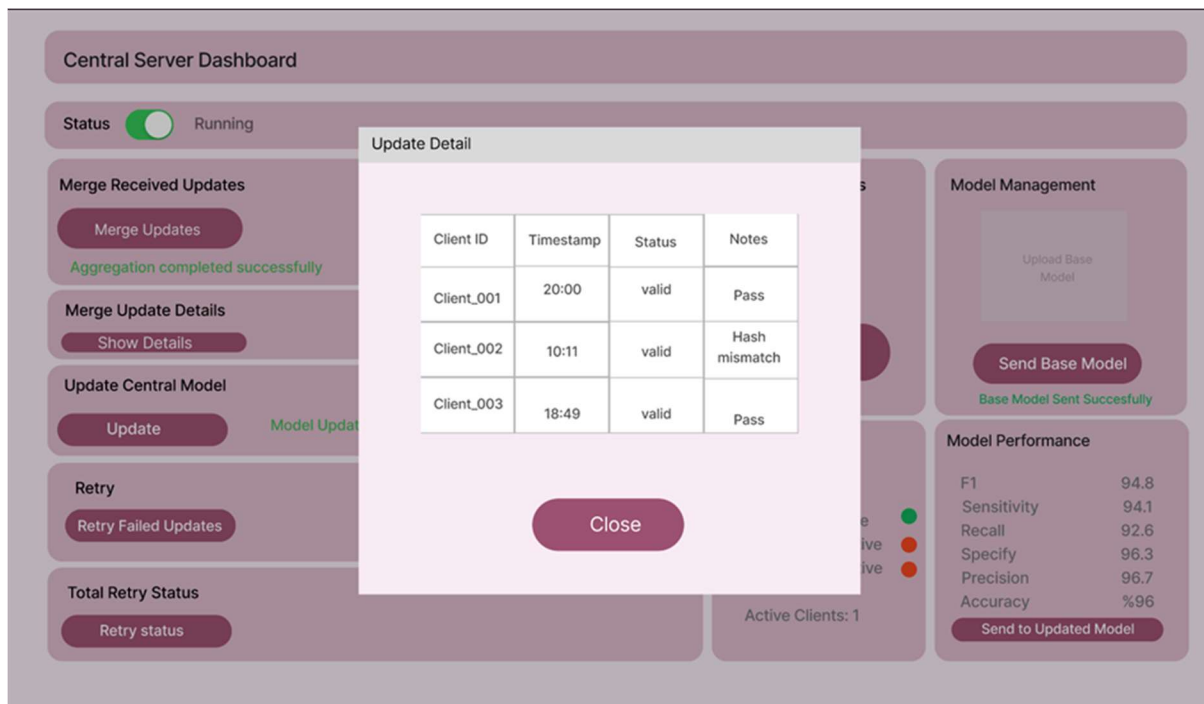


## 6.4 Server UI:

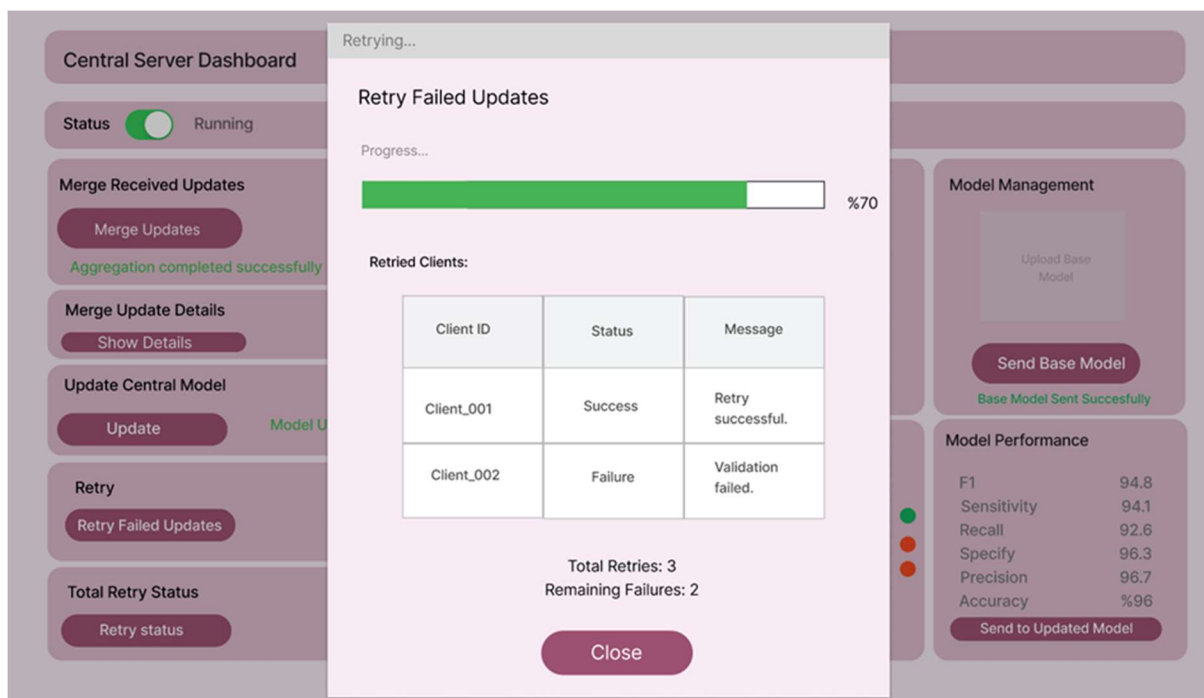
User data stays private as training occurs locally on devices. Only model updates (weights) are shared with a central server, which aggregates them to refine a global model, ensuring privacy and secure collaboration. The process begins with sending the base model to all client devices to initiate federated learning.



Merge Update combines updates received from client devices. This action uses a predefined aggregation algorithm (e.g., weighted average) to unify the updates into a single model update.

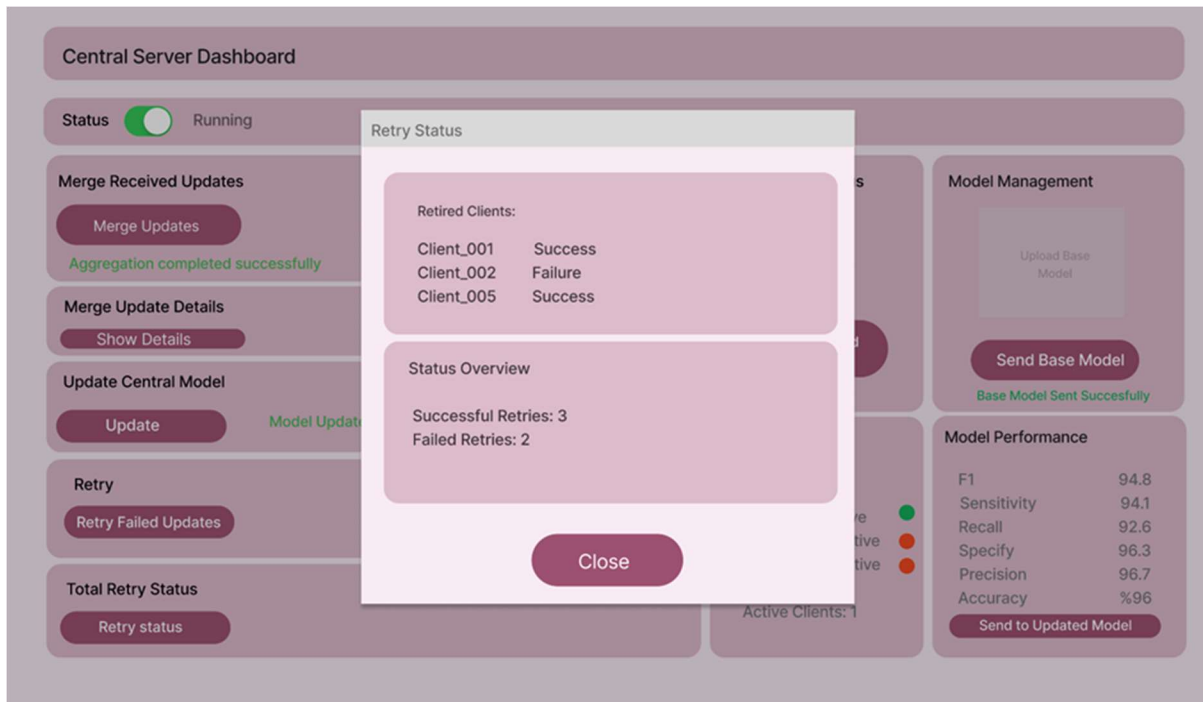


After training locally, client updates are merged, and the central model is improved through a central update. Re-attempts processing for any client updates that previously failed. This ensures all updates are successfully integrated.



Displays a log of past retry attempts, including success rates and any persistent issues encountered.

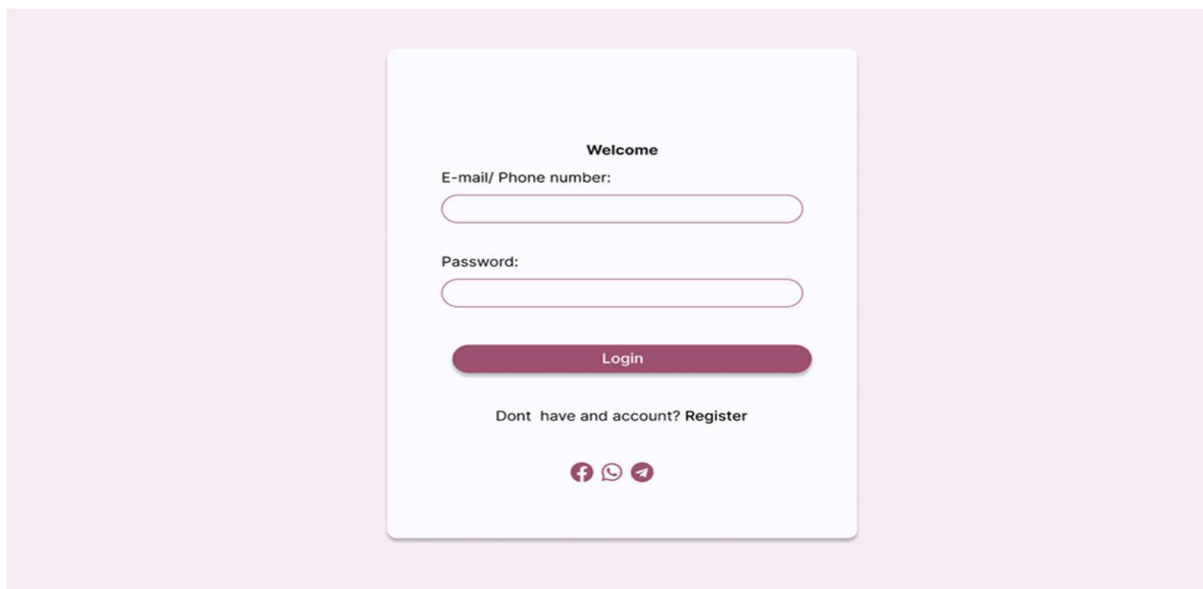






Finally, the updated model is sent back to the clients to ensure synchronization and continued learning.

## 6.5 APPLICATION INTERFACE DESIGN

**Login Section:** The user enters their email or phone number and password, then clicks the "Login" button. The backend validates the credentials against the database. If correct, the user is redirected to the homepage; otherwise, an error message appears.



**Register Section:** The user provides their first name, last name, password, and skin tone, selecting the latter for skin cancer awareness. Upon clicking "Register," the backend checks if the email/phone number is already registered and verifies the password match. If valid, the user is added to the database and redirected to log in; otherwise, an error message appears.



Please Fill out form to Register!

Full name:

Username:

Email/Phone number:

Password:

Confirm Password:


Select Your skin color:

None

Yes i have an account? [Login](#)

Register

**Homepage:** After accessing our homepage, you can view your past results, access your profile and information about us, or upload a new photo to receive the closest diagnosis and explanation.



SKINALYZER


[about us](#)[contact](#)[profile](#)


**SKINALYZER**  
**Skinalyzer: AI-Based Skin Cancer Detection System**

"Skinalyzer" evaluates symptoms on your skin and assists in early disease diagnosis through the power of innovative artificial intelligence (AI) skin technology.

Make sure to consult your doctor without delay.

UPLOAD PHOTO TO ANALYSE

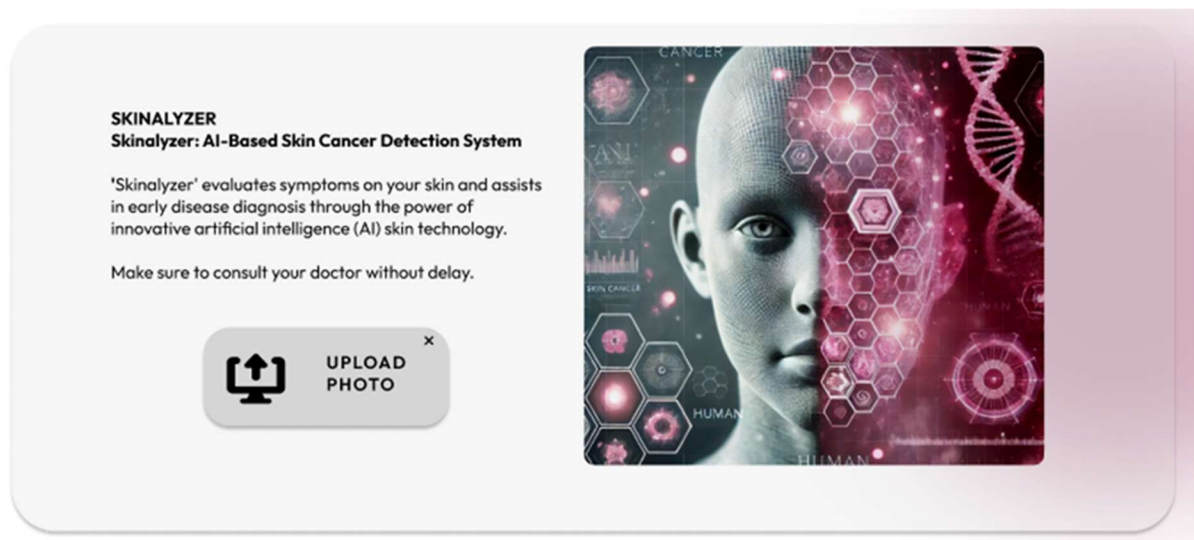




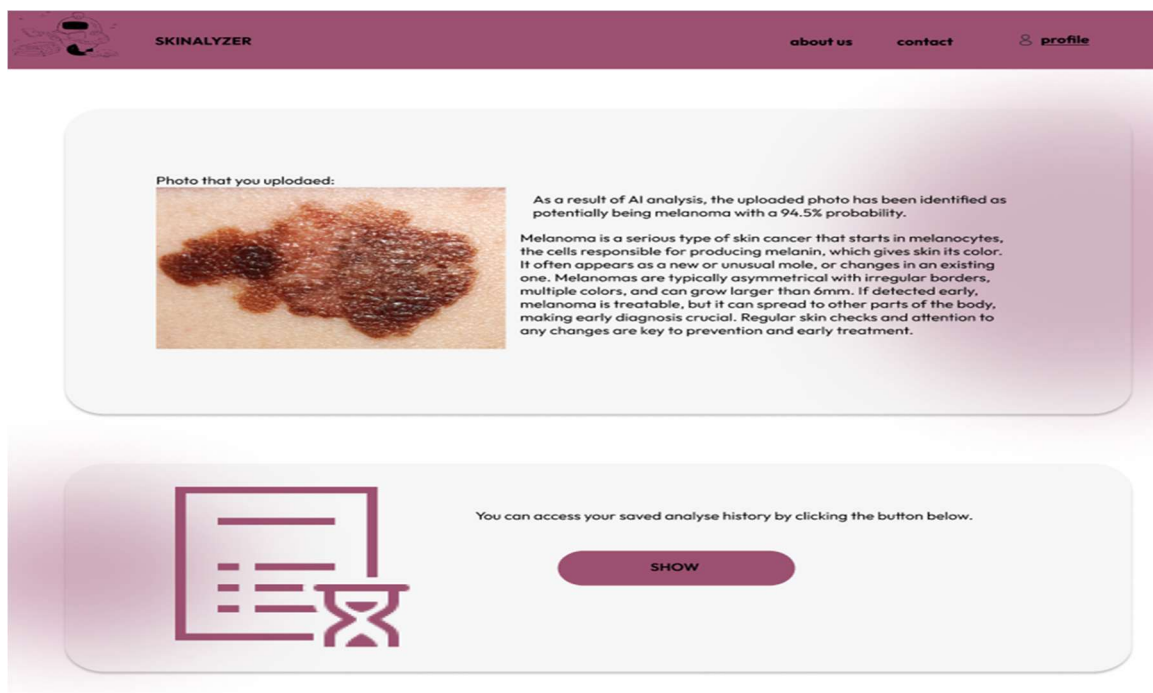
You can access your saved analyse history by clicking the button below.

SHOW

**Upload Photo Button:** By clicking this button, you can drag and drop your photo into the pop-up that appears.

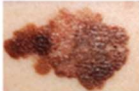


**New Result View:** You can view the uploaded photo, the AI's predicted diagnosis along with its percentage, and the description of the diagnosis. If you wish, you can click the "Show" button below to view all your results

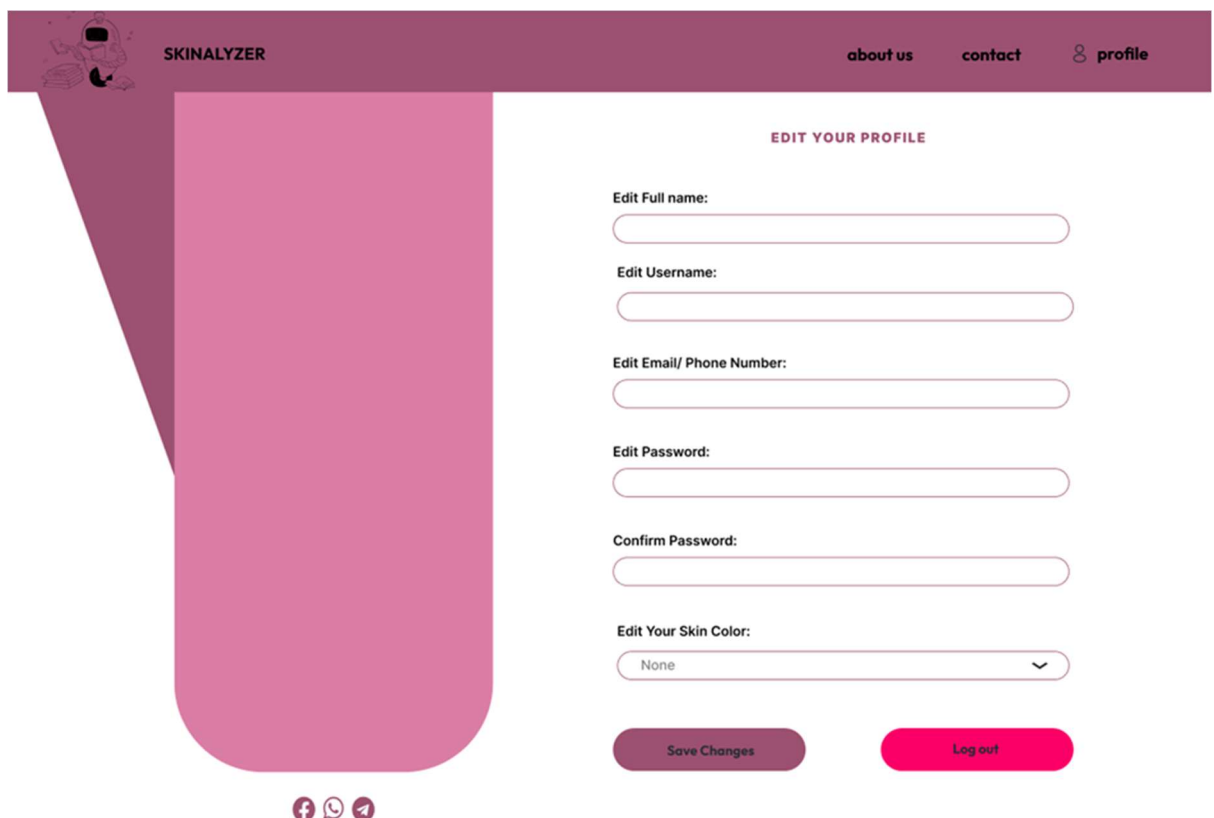


**Show All Result:** On this page, you can find every photo you previously uploaded, along with its diagnosis, percentage, description, and the date it was uploaded.



Your Photo:	Diagnosis:	Definition:	Date:
	Melonama	Melanoma is a serious type of skin cancer that starts in melanocytes, the cells responsible for producing melanin, which gives skin its color.	11.10.2023

**Profile:** Finally, you can access your profile to update the information you provided during registration or log out.

The image shows the "profile" page of the SKINALYZER application. The header bar is identical to the previous image. Below the header, on the left, is a large, vertical, rounded rectangular area with a solid pink background, intended for a profile picture. To the right of this area is a form titled "EDIT YOUR PROFILE". The form contains several input fields: "Edit Full name:", "Edit Username:", "Edit Email/ Phone Number:", "Edit Password:", "Confirm Password:", and "Edit Your Skin Color:". The "Edit Your Skin Color:" field is a dropdown menu currently showing "None". At the bottom of the form are two buttons: "Save Changes" (dark purple) and "Log out" (pink). At the very bottom of the page, centered, are three small circular icons for Facebook, WhatsApp, and Telegram.

## References

- [1] Jang, Bong Kyung, and Yu Rang Park. "Development and Validation of Adaptable Skin Cancer Classification System Using Dynamically Expandable Representation." *Healthcare Informatics Research* 30.2 (2024): 140-146.
- [2] Iqbal, Saeed, et al. "AMIAC: adaptive medical image analyzes and classification, a robust self-learning framework." *Neural Computing and Applications* (2023): 1-29.  
Gottumukkala, VSSP Raju, N. Kumaran, and V. Chandra Sekhar. "BLSNet: Skin lesion detection and classification using broad learning system with incremental learning algorithm." *Expert Systems* 39.9 (2022): e12938.
- [3] Luo, Yong, et al. "An appraisal of incremental learning methods." *Entropy* 22.11 (2020): 1190.
- [4] Lomonaco, Vincenzo, and Davide Maltoni. "Comparing incremental learning strategies for convolutional neural networks." *Artificial Neural Networks in Pattern Recognition: 7th IAPR TC3 Workshop, ANNPR 2016, Ulm, Germany, September 28–30, 2016, Proceedings 7*. Springer International Publishing, 2016.
- [5] NERGİZ, M. "Collaborative Artificial Intelligence Concept: Federated Learning Review." *DÜMF Mühendislik Derg* (2022).  
Yaqoob, Muhammad Mateen, et al. "Federated machine learning for skin lesion diagnosis: an asynchronous and weighted approach." *Diagnostics* 13.11 (2023): 1964.  
Riaz, Shafia, et al. "Federated and Transfer Learning Methods for the Classification of Melanoma and Nonmelanoma Skin Cancers: A Prospective Study." *Sensors* 23.20 (2023): 8457.
- [7] Sumaiya, Noor, and Anooja Ali. "Federated Learning Assisted Deep learning methods fostered Skin Cancer Detection: A Survey." *Frontiers in Biomedical Technologies* (2024).
- [8] Hashmani, Manzoor Ahmed, et al. "An adaptive federated machine learning-based intelligent system for skin disease detection: A step toward an intelligent dermoscopy device." *Applied Sciences* 11.5 (2021): 2145.
- [9] Ayromlou, Sana. *Incremental learning and federated learning for heterogeneous medical image analysis*. Diss. University of British Columbia, 2023.

<https://www.innova.com.tr/blog/Yapay-zeka-ile-saglik-sektorunde-hastalik-teshisinde-devrim>

**ISIC 2019 Dataset:** <https://challenge.isic-archive.com>

**HAM10000 Dataset:** <https://www.kaggle.com/kmader/skin-cancer-mnist-ham10000>

**Incremental Learning and Federated Learning for Heterogeneous Medical Image Analysis:** Incremental learning and federated learning for heterogeneous medical image analysis - UBC Library Open Collections