**Class Diagram:**

@startuml

left to right direction

skinparam classAttributeIconSize 0

hide circle

hide stereotypes

class User {

-userID: int

-name: string

-email: string

-password: string

-role: string

--

+register()

+login()

+logout()

+viewPastRecords()

}

class Image {

-imageID: int

-userID: int

-uploadDate: datetime

-imagePath: string

--

+uploadImage()

+deleteImage()

+getImageDetails()

}

class Prediction {

-predictionID: int

-imageID: int

-modelID: int

-predictedDisease: string

-confidenceScore: float

-predictionDate: datetime

--

+runPrediction()

+getTopPredictions()

}

class Model {

-modelID: int

-modelName: string

-version: string

-trainedDate: datetime

--

+loadModel()

+classifyImage(image: Image)

+updateModel()

}

class DiseaseInfo {

-diseaseName: string

-description: string

-referenceURL: string

--

+getInfo()

+updateInfo()

}

' Persistence/service layer to ensure SQL saving of predictions

class PredictionRepository {

+save(p: Prediction)

+findByImage(imageID: int): Prediction

+findByUser(userID: int): List<Prediction>

}

class SQLPredictionRepository {

+save(p: Prediction)

+findByImage(imageID: int): Prediction

+findByUser(userID: int): List<Prediction>

}

class PredictionService {

+predictAndSave(imageID: int): Prediction

+getPredictionsByUser(userID: int): List<Prediction>

}

' Domain relationships

User "1" o-- "0..\*" Image : uploads

Image "1" \*-- "1" Prediction : generates

Prediction "0..\*" --> "1" Model : uses

Prediction "0..\*" --> "1" DiseaseInfo : refers to

' Service/Repository wiring for SQL persistence

PredictionService --> Model : uses

PredictionService --> PredictionRepository : persists

PredictionRepository --> Prediction : manages

SQLPredictionRepository ..|> PredictionRepository

' Layout nudges

User -[hidden]-> Image

Image -[hidden]-> Prediction

Prediction -[hidden]-> DiseaseInfo

Prediction -[hidden]-> Model

PredictionRepository -[hidden]-> SQLPredictionRepository

@enduml

**Activity Diagram:**

@startuml

' Error-free compact activity diagram (no left-to-right, no goto)

skinparam shadowing false

skinparam ArrowColor #333333

skinparam ActivityBorderColor #666666

skinparam ActivityBackgroundColor #EEF3F7

skinparam ActivityFontColor #1F2D3D

skinparam RoundCorner 10

title Skin Disease Detection - Compact Activity Diagram

start

:Open app;

if (Authenticated?) then (yes)

else (no)

:Register or Login;

if (Register?) then (yes)

:Enter name, email, password, role;

:Validate inputs;

:Hash + salt password;

:Create account;

endif

:Enter email + password;

:Verify credentials;

if (Auth OK?) then (yes)

else (no)

:Show "Not authenticated";

stop

endif

endif

:Show role-based dashboard (Patient / Medical);

if (View history?) then (yes)

:Authorize access to user's records;

:Fetch past uploads & predictions by user;

:Display history (image, disease, confidence, date);

:Back to dashboard;

else (Upload image)

endif

' Upload + validation + preprocessing (repeat loop; exits with break on success)

repeat

:Choose source (Local / Webcam);

:Get image or cancel;

if (Canceled?) then (yes)

:User canceled;

stop

endif

:Validate format (JPEG, PNG);

if (Format valid?) then (yes)

:Preprocess (resize, normalize, prepare tensor);

if (Preprocessing OK?) then (yes)

break

else (no)

:Show "Preprocessing failed";

endif

else (no)

:Show "Incorrect image format";

endif

repeat while (Retry?)

:Select model (CNN / MobileNet / DenseNet121);

:Classify -> top-k predictions with confidences;

if (Prediction OK?) then (yes)

else (no)

:Show "Prediction failed";

stop

endif

fork

:Save to DB:\n- image metadata (userId, path, date)\n- prediction (modelId, labels, confidences, timestamp)\n- enforce access control & secure storage;

fork again

:Fetch disease details + reference URLs for top predictions;

end fork

:Show results (chart/table of probabilities);

:Open disease details or external links;

:Actions: View History | Upload Another | Dashboard;

stop

@enduml

**Sequence Diagram**

@startuml

' Compact, aggregated sequence diagram (few lifelines, minimal steps)

skinparam shadowing false

skinparam ArrowColor #333333

skinparam ParticipantBorderColor #666666

skinparam ParticipantBackgroundColor #EEF3F7

skinparam ParticipantFontColor #1F2D3D

skinparam NoteBackgroundColor #F9FBFC

skinparam NoteBorderColor #999999

title Skin Disease Detection - Aggregated Sequence Diagram

actor "User\n(Patient or Medical)" as User

participant "System (UI/API)" as Sys

database "Database" as DB

participant "AI Model\n(CNN/MobileNet/DenseNet121)" as AI

participant "Disease Info" as Info

== 1) User Login ==

User -> Sys: 1. Login(credentials)

Sys -> DB: Get account by email (returns stored\_hash)

Sys -> Sys: check\_password\_hash(stored\_hash, password)

alt Valid credentials

Sys -> DB: Create session

Sys --> User: Access granted (dashboard)

else Invalid credentials

Sys --> User: Error + prompt to re-login

end

note over Sys

Using werkzeug.security:

- generate\_password\_hash() on signup

- check\_password\_hash() on login

end note

== 2) Image Analysis (Upload + Validate + Process) ==

User -> Sys: 2. UploadImage(source: local/webcam)

Sys -> Sys: Validate format & size

alt Invalid image

Sys --> User: Upload error (unsupported/too large)

else Valid image

Sys -> Sys: Preprocess (resize, normalize)

alt Preprocessing OK

Sys -> AI: 3. Classify(preprocessed image)

AI --> Sys: Predictions + confidence scores

Sys -> DB: Store results (metadata, scores, model, timestamp)

Sys -> Info: Fetch reference links

Info --> Sys: URLs

Sys --> User: Show results (charts + details + links)

else Preprocessing failed

Sys --> User: Preprocessing error

end

end

== 3) Past Records ==

User -> Sys: 4. GetHistory()

Sys -> DB: Fetch records by userId

DB --> Sys: Images + diseases + dates + scores

Sys --> User: Display history

@enduml

**Component Diagram**

@startuml

' Compact, A4-friendly component diagram

' Tip: adjust scale (0.60–0.85) to fit your page layout.

scale 0.72

left to right direction

skinparam shadowing false

skinparam ArrowColor #333333

skinparam ArrowThickness 1

skinparam componentStyle rectangle

skinparam defaultFontSize 12

skinparam component {

BackgroundColor #EEF3F7

BorderColor #666666

FontColor #1F2D3D

}

title Skin Disease Detection - Component Diagram (Compact)

package "Frontend" {

[Web Interface\n(upload • results • dashboard)] as Web

[Result Display\n(charts • probabilities)] as Results

Web ..> Results : render

}

package "Backend / API" {

[API Layer\n(request routing • validation)] as API

[Auth (Werkzeug)] as Auth

[Logging / Errors] as Log

database "Database\n(users • images • predictions)" as DB

}

package "ML Pipeline" {

[Preprocess\n(resize • normalize • augment)] as Prep

[AI Model\n(CNN | MobileNet | DenseNet121)] as Model

}

[Medical References] as Info

' Core interactions (minimal labels to keep compact)

Web --> API : REST (JSON/multipart)

API --> Auth : login/signup\n(generate\_password\_hash /\ncheck\_password\_hash)

API --> Prep : image

Prep --> Model: tensor

Model --> API : predictions

API --> DB : store/fetch

API --> Info : refs for diseases

API --> Web : results JSON

API ..> Log : events + errors

@enduml

**Architecture**

@startuml

title AI For Skin Disease Detection Architecture (Password Auth)

skinparam componentStyle rectangle

skinparam shadowing false

skinparam dpi 220

skinparam defaultFontSize 14

top to bottom direction

scale 780 width

actor User

rectangle "Client" as Client {

component "UI\n(Upload/Webcam,\nSign-in)" as UI

}

rectangle "Backend" as Server {

component "Auth\n(Password Login)" as Auth

component "Inference\n(Preprocess + Route + Predict)" as Inference

component "Result\n(Prediction + Confidence)" as Result

}

rectangle "Models" as Models {

component "Best-Model\nSelector" as Router

together {

component "DenseNet121" as DenseNet

component "MobileNet" as MobileNet

component "Custom CNN" as CustomCNN

}

}

rectangle "ML Pipeline" as MLPipe {

component "Training\n(Augment • Fit • Tune)" as Train

component "Evaluation\n(Metrics)" as Eval

component "Registry\n(Best Model)" as Registry

}

database "Storage\n(Data • Checkpoints • Reports)" as Store

' Inference (runtime)

User --> UI

UI --> Auth : sign in (username + password)

Auth --> UI : success / failure

UI --> Inference : image

Inference --> Router

Router --> DenseNet

Router --> MobileNet

Router --> CustomCNN

DenseNet --> Result

MobileNet --> Result

CustomCNN --> Result

Result --> UI

UI --> User : prediction + confidence

' Training & evaluation (offline, dashed)

Store ..> Train : read/write

Train ..> Eval : results

Eval ..> Registry : metrics

Train ..> Store : checkpoints

Registry ..> Store : best weights

Registry ..> Router : deploy best

Store ..> Inference : load model

note bottom of Auth

Password-only authentication

(no tokens used)

end note

note bottom of Train

Augmentation is used only during training

end note

@enduml