Detailed Design

Architecture

The architecture that best suits our project is Client-Server.

Client-Server Architecture in our project:

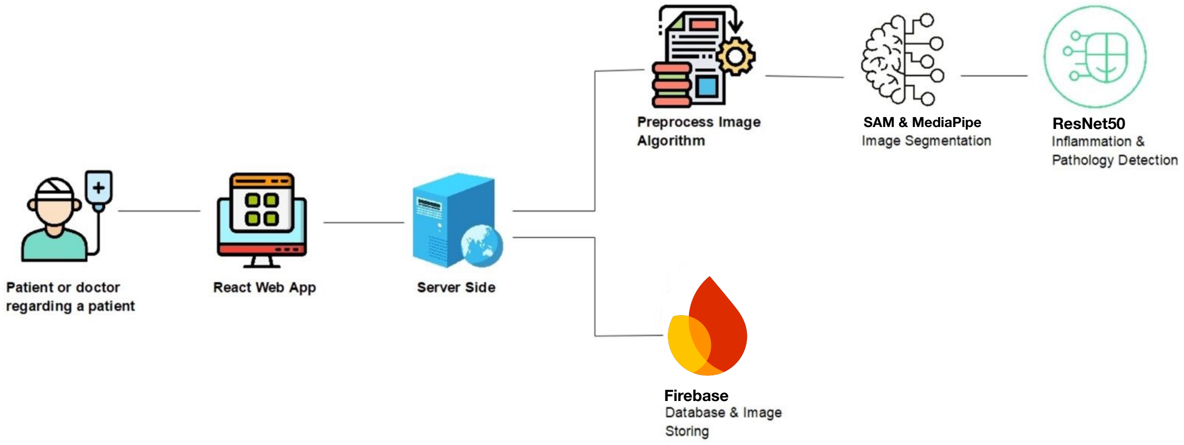
- Server:
  - ❖ Handles server-side logic via Firebase Functions (serverless backend).
  - ❖ Communicates with a machine learning inference model hosted on HuggingFace.
  - ❖ Stores and retrieves data using Firebase Realtime Database and Firebase Cloud Storage.

- Client:
  - ❖ A React-based web application that uploads images for processing and displays results.
  - ❖ Interacts with Firebase backend and model endpoints to visualize predictions and results.

Data Storage

- User metadata is stored in Firebase Realtime Database.
- Images are stored in Firebase Cloud Storage.
- All storage is secured and linked to authenticated user sessions.

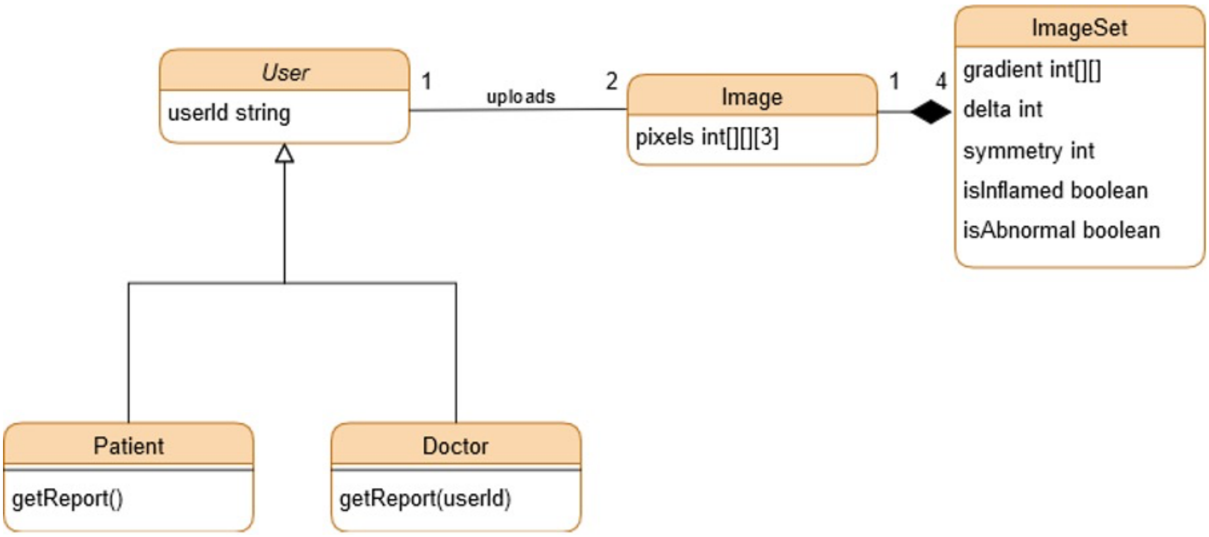
Graphic Description


Preprocess Image Algorithm
SAM & MediaPipe
Image Segmentation
ResNet50
Inflammation & Pathology Detection
Patient or doctor regarding a patient
React Web App
Server Side
Firebase
Database & Image Storing

Data Description

- User – a JSON file that has the following fields:
  - ❖ userID
- Image sets' data – will be saved as JSON with the following fields:
  - ❖ Delta
  - ❖ Gradient
  - ❖ Symmetry
  - ❖ Any other data that we will decide in the future is relevant for the model.

Graphic Description



API specification

- Upload optical and thermal images of hands.
- Display prediction results of the ML model.
- Upload the images to the database for training the model.
- Trigger the full preprocessing and classification pipeline (segmentation, landmark detection, and joint prediction).
- Retrieve session results per user (view previous predictions and image data).

Programming languages and tools

- Python for preprocessing, segmentation, and model inference.
- React.js for frontend development due to simplicity and dynamic rendering.
- Firebase Functions for backend orchestration.
- Firebase Realtime Database and Firebase Cloud Storage for data persistence.

Algorithm Description

Hand Registering Algorithm:

Algorithm: *The registration process aligns thermal and optical images of the same hand using landmark-based geometric transformation. The pipeline begins with MediaPipe, which detects 21 anatomical landmarks on the hand in the optical image. The same hand is segmented using SAM (Segment Anything Model) to isolate it from the background.*

If a matching thermal image exists, the thermal hand mask is generated using the corresponding SAM segmentation. The landmarks from the optical image are geometrically transferred to the thermal image space using the overlapping hand region as a guide.

To evaluate alignment accuracy, the system calculates the Euclidean distance between corresponding landmarks in the optical and thermal domains. If the average landmark distance exceeds a defined threshold (e.g., 10 pixels), the registration is considered invalid, and the image is discarded from further processing.

This method ensures spatial consistency without relying on fragile keypoint descriptors like ORB or SURF, which often fail on low-contrast thermal images.

Time Complexity:

- **Landmark Detection (MediaPipe):** O(n), where *n* is the number of pixels (real-time optimized).
- **Segmentation (SAM):** O(n), transformer-based attention model.
- **Landmark Transfer + Distance Evaluation:** O(L), where *L = 21* is the fixed number of landmarks.
- **Overall:** O(n), linear with respect to image size. Efficient in practice due to fixed input dimensions and optimized model inference.

Machine Learning Models:

1. **MediaPipe – Hand Landmark Detection:**

   Algorithm: *MediaPipe Hands is a machine learning pipeline by Google that detects and tracks 21 3D hand landmarks in real time. It uses a palm detection model followed by a regression model that outputs precise landmark positions. The landmarks represent key joint positions of the fingers and palm.*

   Time Complexity: The runtime is approximately O(n) with respect to the number of image pixels, but practically optimized for real-time performance on both CPU and GPU.

2. **SAM – Segment Anything Model:**

   Algorithm: *SAM is a transformer-based model trained to generate segmentation masks for any object in an image, given a prompt such as a bounding box or point. In this project, SAM is used to segment the hand region from optical images to isolate the relevant area for further analysis.*

   Time Complexity: The model's complexity depends on the number of prompts and image resolution. For a single point or bounding box input, inference is approximately O(n), where n is the number of pixels. Internally, the attention mechanisms scale based on patch count, but optimizations make inference tractable.

3. ResNet50 - Joint-level Inflammation Classification

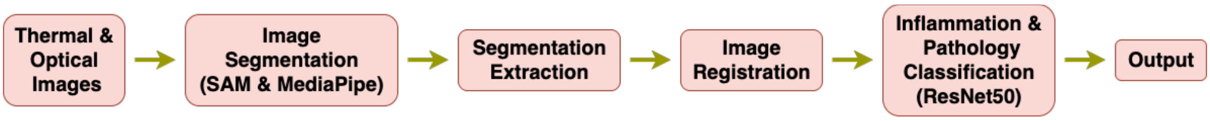
Algorithm: *ResNet50 is a 50-layer deep convolutional neural network designed with residual blocks to enable efficient training of very deep models. In this project, 32 separate ResNet50 models were trained — one for each joint (excluding fingertips) — to classify whether inflammation is present based on a 4-channel input patch.*

Each input includes:

- *The registered thermal image*
- *A binary mask of the hand*
- *A masked thermal map*
- *A palm-centered distance map*

Time Complexity: The forward pass of ResNet50 has a complexity of approximately O(n·d²), where n is the number of pixels in the input patch and d is the feature map dimension. However, since the input patches are small and fixed in size, inference remains fast (typically <1s per joint on CPU).

Graphic Description



Thermal & Optical Images → Image Segmentation (SAM & MediaPipe) → Segmentation Extraction → Image Registration → Inflammation & Pathology Classification (ResNet50) → Output