# FitLens - Documentation

# Overview

FitLens is an AI-powered measurement system using YOLOv8n segmentation, MediaPipe landmarks, and pixel-to-scale conversion and returns body measurements

## Flow Chart

A screenshot of a diagram

AI-generated content may be incorrect.

## Processing Pipeline

## - Preprocessing

## - YOLOv8n-Seg masking

## - MediaPipe landmark detection

## - Pixel-to-scale measurement

## - Result packaging

**Database Schema**

Users(id, email)

Sessions(id, user\_id, status)

Captures(id, session\_id, view\_label, image\_urls)

Measurements(id, session\_id, measurement\_json)

A screenshot of a computer

AI-generated content may be incorrect.

## System Architecture Overview

## A screenshot of a computer AI-generated content may be incorrect.

**Activity Diagram**

A diagram of body measurement system

AI-generated content may be incorrect.

## Wireframes

-Home Screen

- Live Camera Mode Screen

- Results Screen

**UI Mockups – Home Screen**

Home Screen - Choose Upload or Live Camera

A screenshot of a cell phone

AI-generated content may be incorrect.

**Upload Mode**

- Front and Side upload inputs and height field

A screenshot of a computer

AI-generated content may be incorrect.

**Measurement Results**

- Landmarks & Segmentation Mask (Front View , Side View)

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a body scan

AI-generated content may be incorrect.

**Measurement Results**

- Measurements list and table

A screenshot of a computer

AI-generated content may be incorrect.

## UML Diagrams

A screenshot of a computer

AI-generated content may be incorrect.

**Sequence Diagram**

A screenshot of a computer

AI-generated content may be incorrect.

Developer Handbook (Summary)

This section provides actionable guidance for developers implementing the FitLens system. It covers the backend API contracts, worker pipeline steps, data model, deployment notes, and testing checklist.

Key items:

- API Endpoints: sessions, captures, results, auth (JWT).

- Worker Pipeline: image pre-processing, YOLOv8n-Seg inference, MediaPipe landmark extraction, pixel-to-scale conversion, result packaging.

- Storage: Use S3-compatible storage for images and masks; store metadata and measurements in PostgreSQL.

- Queue: Use Redis/RabbitMQ + Celery/RQ for job orchestration.

- Deployment: GPU-enabled workers (CUDA), Kubernetes or ECS for scaling, monitoring with Prometheus/Grafana.