**A logo for a university

Description automatically generated**

**Title: Webcam Application for High-Risk Independent Resident**

**Author: Tianluan Lin**

***Course:* Computer Science**

**Project code: PJE40**

**Supervisor: Ioannis Kagalidis**

**May 2025**

Please tick

|  |  |
| --- | --- |
| ❏ | I give permission for my project to be published in the University library and/or be made available to other students as examples of previous work. (optional). |
| ❏ | I confirm that I have read and understood the University Rules in respect of plagiarism and student misconduct. |
| ❏ | I declare that this work is entirely my own. Each quotation or contribution cited from other work is fully referenced. |
|  |  |

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

This display the development of a webcam application that are designed to monitor the resident under high risk or other venerable situations which could lead to injury or potential hazards. It will combine with local weather information support and Convernutionary Neuron Network (CNN) for image recognition to help user to identify the situation and status of high-risk residents.

This report will provided full process of its development and the theory of key elements. Although the project partially achieved many of the design goal for a monitor application, this project provided a clear idea of what are the key elements for a monitor application for this type of tasks. It provides a potential pathway for a comprehensive and a mature monitor application for the smart devices and IoT.

# Acknowledgements

# Table of Contents

# Chapter 1 – Introduction

## 1.1 Background

Traditional healthcare and monitoring systems have long relied on human supervision and warden. Even in professional environments such as nursing homes, constant individual supervision is not always feasible. Furthermore, there have been reported cases of neglect or even abuse in residential care facilities. On the other hand, many high-risk individuals, including elderly people, live alone without immediate access to assistance. In the UK alone, there are 3.3 million pensioners who live independently (Office for National Statistics, 2023) and are at risk of possible injuries or unconsciousness at any moment. The possibility of such an event would only increase with age. Delays in medical response due to undetected falls can lead to severe health deterioration or even death (WHO, 2021). Moreover, as extreme weather conditions become more frequent than ever before, vulnerable individuals who live on their own may refuse to use heating or cooling appliances, often due to cost or personal habits. This has caused casualties of illnesses related to temperatures like hypothermia or dehydration every year.

The demand for technologies that assist high-risk individuals who are living independently is increasing with more reliable and more powerful tools and equipment as the technologies have improved in terms of computational power and networking for the past decades. This project aims to address that need by developing a prototype webcam monitor application system for detecting whether a person has fallen using computer vision techniques and machine learning. By using a camera, the system captures images at a rate of two frames per second, processes them using a Convolutional Neural Network (CNN), and classifies the user’s pose to determine whether a fall has occurred.

If a fall is detected, the system will automatically notify a pre-configured contact, enabling a timely response. Falls are one of the leading causes of injury-related death among adults in many countries. Additionally, another module provides health-related feedback based on temperature and humidity, helping ensure that the resident is living in safe conditions.

## 1.2 Project Aims and Objectives

The core objective is to design and implement a real-time fall detection system using images captured by a webcam and analysed via a CNN-based pose estimation model. The secondary objective is to provide environmental advice based on sensor data.

Primary Objectives:

* Capture webcam images at a consistent frame rate.
* Train and deploy a CNN to classify human poses.
* Determine if a set of poses indicates a fall.
* When a fall is detected, pass the message to relevant contactor.

Secondary Aspirations: (Implementation subject to time and complexity constraints)

* Provide a text-based advisory system based on environmental conditions.
* Record daily activity time
* Log medication intake times.
* Deliver personalized alerts or reminders based on behavioural patterns.

## 1.3 Scope and Constraints

This project is developed under academic constraints and does not extend to clinical or regulatory compliance for medical-grade devices. The system is intended as a prototype to demonstrate feasibility rather than a commercialised product. While the application is designed to operate on consumer-grade hardware (laptop and webcam), and is not designed for multi-camera setups or advanced motion tracking.

Due to the incident during training of the model, this application is tested and trained on a CPU-based system. Hence, the CNN model complexity and the speed of image processing are limited.

## 1.4 Ethics Review

This monitoring system does not store or process personally identifiable information beyond what is necessary for fall detection and alert generation. The project has received ethics approval (Reference: TETHIC-2025-110551) from the Technology Faculty Ethics Committee, confirming that it involves no physical, psychological, or environmental risks.

## 1.5 Report Structure

This report is structured as follows:

Chapter 2 - Literature Review: Critical analysis of relevant studies in the fall detection, pose estimation, and ambient health monitoring.

Chapter 3 – Methodology: Justification of the development approach, neural network design, and system structure.

Chapter 4 – Requirements: Functional and nonfunctional requirements.

Chapter 5 – Design: Detailed system architecture, data flow, and model selection.

Chapter 6 – Implementation: Description of key components including webcam interfaces, CNN training, and alert mechanism.

Chapter 7 – Testing and Evaluation: Verification of results against requirements and performance benchmarks.

Chapter 8 – Conclusion and Future Work: Summary of findings and potential improvements.

# Chapter 2 – Literature Review

## 2.1 Introduction

This chapter presents a critical review of the technologies and techniques that support fall detection in high-risk individuals, particularly those living alone. The main areas of discussion include:

1. Sensor-based fall detection mechanisms – List the current fall detections and their disadvantages.
2. Pose estimation through computer vision – Discuss a few popular approaches for getting the pose of a person from the image.
3. The use of CNNs in human activity recognition – Discuss what is CNN and how will this monitor application system’s CNN work.
4. The effect of the environmental monitoring – Why does the monitor application also need to evaluate the environment for high-risk individuals.

Outline of each technique of these topics will be discussed and the key points and basic requirements of this monitor application will be drawn. The limitation of existing systems will also be discussed, leading to rationale for using RGB image-based CNN classification in the proposed solution.

## 2.2 Fall Detection in High-Risk Residents

Falls are one the most serious health threats to older adults, commonly resulting in fractures, trauma, or ling-term hospitalisation. According to the U.S. Centres for Disease Control and Prevention (CDC), falls are a leading cause of traumatic brain injuries and prolonged recovery demands on healthcare systems (CDC, 2019).

Currently, there are hardware devices have been developed or under developing to detect fall of a person. These fall detection method each has its own advantages and disadvantages.

* Wearable Sensors: Devices which are designed to be worn on the residents on different part of the body. Some are just like watches on the arm, some of them need to be placed on the main body. These types of devices using accelerometer to identified the motion and the high-risk residents’ condition. However, these devices must be worn at all time and can produce inaccurate results if the devices are worn at different places or shift position on the body (Singh et al., 2020). This limit the place and scenario of the device being used.
* Ambient Sensors: This type of devices collect the information from the environment to identifies falls.
* Acoustic Sensors: Classify the sound of fall as well as other sounds from the surrounding. However, this type of device’s accuracy is affected by background noise. Even the presents of large pets will lead to inaccurate results.
* Floor Pressure Sensors: This type of sensor is still under development in the lab environment. It detects vibrations from the fall impact. However, the performance varies based on floor material and body weight. Another approach is to put the pressure sensor around the bed to detect the fall around the bed only (Huan-Wen Tzeng, 2010). This type of fall detection has very limit of use in terms of the area it will be able to cover.
* Infrared, Radar, Ultrasonic Sensors: These types of sensors offer contactless monitoring by placing them in a carefully selected place. However, these require precise setup and specialised equipment that are not typically common in households.
* Image Sensors: These use cameras to visually detect falls. While accurate, many reply on cameras with special functions like 3D camera or depth camera, which are expensive and not typically common to our ordinary life. Additionally, privacy concerns arise when capturing live video streams or photos has risk of leak to the social media.

[Insert Table 1: Comparison of Fall Detection Technologies – Sensor Type, Cost, Accuracy, Limitations, Real-world Usability]

## 2.3 Computer Vision and Pose Estimation Techniques

Pose estimation aims to localise key body joints in images or videos. Popular frameworks such as OpenPose, BlazePose, and MediaPipe are capable of reconstructing a full human skeleton for motion analysis. These models often use multiple camera angles or RGB-D inputs to infer pose in 2D or 3D space. However, most state-of-the-art pose estimation frameworks have high demands for hardware requirements like depth data or high-resolution camera. Otherwise, technical issues like occlusion or low-light would occur leads to incorrect body skeleton model (Mehta et al., 2016). These will also demand more computational power for a real-time application like our monitor application.

[Insert Figure 2.3.1: Example of keypoint detection in OpenPose and RGB-only visual classification]

Nonetheless, existing pose estimation models/software are mutual and many of them have a high accuracy rate than typical CNNs-based pose estimation. This project will select Y

## 2.4 CNNs in Human Activity Recognition

Convolutional Neural Network (CNN) is a class of deep learning models designed for image-based pattern recognition. CNNs have been widely adopted in human activity recognition (HAR) due to their ability to detect statical features from raw image data. A typical CNN architecture includes:

* Input Layer: Fixed-size RGB image will be received at this layer at beginning.
* Convolutional Layers: This layer applies filters to extract visual features such as edges, shapes, or patterns.
* Activation Layers: Using functions to introduce non-linearity into the model. Typical activation functions include ReLU, SoftMax, and Tanh etc.

[ Insert Figure2.4.1: Different activation functions]

* Pooling Layers: This layer reducing the sampling rate to reduce overfitting and training time. Two typical pooling layers are Max Pooling and average Pooling.
* Fully Connected Layers (Output Layers): This layer will combine features to classify the input image.

Currently, many HAR software use CNN as a primary structure to build their models. Back to 2016, researches in this filed has already developed a “binary sensor network” enhanced by CNNs. It was able to identify daily human activities that claimed accuracy rate of at least 94.7% with test samples (Liu et al., 2016)

In the monitor application system, we will be trying to train a CNN for HAR with a typical CNN structure. However, due to hardware constrain, a degraded, simpler CNN with less channels and convolutional layers were considered. Given the limitations with the custom CNN model, the monitor system will adopt the pre-trained YOLO11n-pose model. This model is developed by Ultralytics was able to detect 17 human key points and make pose estimation based on the key points (Ultralytics). Allowing our monitor application system to have a more reliable core HAR system to achieve the primary objective.

[Insert Table 2.4.2: CNN Architecture – Layer Name, Filter Size, Output Shape, Activation Function]

## 2.5 Environmental Sensing in Elderly Monitoring

Temperature and humidity are critical for elderly welling. High indoor temperatures are associated with heat exhaustion and dehydration, while cold environments increase the risk of hypothermia (WHO, 2018). Smart monitoring systems often include sensors that trigger alerts when ambient conditions fall outside safe thresholds. In this project, the monitor application includes a secondary module that provides text-based advice derived from local environmental data. This lightweight approach is suitable for resource-constrained systems and avoids the need for expensive multi-sensor arrays.

[Insert Figure 2.5.1: Diagram of temperature/humidity-based advisory logic used in the system]

## 2.6 Research Gaps and Design Justification

Many existing fall detection systems have achieved high accuracy. Yet, there are still limitations in each of those systems:

1. Sensor-heavy.
2. Hardware specified for a particular detection method.
3. Low cost-effective.
4. Rarely integrated.

Adhikari et al. (2017) proposed a CNN-based method using both RGB and depth data for indoor fall detection. However, their system depended on Kinect-like devices and requirement for depth camera. Inspired by his team’s research, this project seeks to replicate its effectiveness using only readily accessible RGB webcams. The aim of this project is to develop a prototype of a system that covers simplicity, affordability, and extensibility that other fall detection application systems unable to achieve.

## 2.7 Summary

This chapter reviewed the evolution of fall detection technologies, pose estimation models, and CNNs-based image recognition. Key design decisions, such as using RGB only inputs and avoiding expensive skeleton estimation models and depth camera, are justified through both technical and practical considerations. The literature supports a strong case for the development of a light-weight, real-time, webcam-based solutions to monitor vulnerable individuals living independently.

# Chapter 3 – Methodology

# 

## 3.1 Introduction

This chapter outlines the methodology and approaches to developing the Monitor Application for High-Risk Independent Residents. The system’s core functionality revolves around real-time webcam-based image capture and pose detection. Initially, a custom CNN-based pose detection model was developed. However, due to its low testing accuracy (~33%) and high validation loss (~1.2), this approach was replaced with a pre-trained pose estimation model from Ultralytics YOLO. This transition ensured a more robust and dependable core for the system.

## 3.2 Overall System Development Approach

The core of the monitor application, pose estimation is divided into 3 subsystems: real-time webcam feed, pose estimation and fetch pose to webpage. The application was built using an interactive development methodology for software development. This modular structure allowed for parallel development and the ability to substitute components if technical issues arose, such as the switch from a custom CNN to a pre-trained model.

## 3.3 Data Acquisition and Preprocessing

### 3.3.1 Data Source

The application captures real-time data from a standard consumer-grade webcam. Due to hardware constraints and a focus on accessibility, the system captures images at 2 frames per second (FPS) by taking screenshots rather than streaming video continuously. During testing, frame rates above 4 FPS led to corrupted “snowy picture” displays on the web interface. Although the root cause remains unclear, reducing the capture rate significantly mitigated the issue. Each frame is saved locally and processed for pose estimation.

### 3.3.2 Preprocessing Steps

Captured images are pre-processed before being passed to the pose estimation model module:

* Resizing: Each frame will be resized to 640x640 pixels to match the input for the pose estimation model. A typical webcam resolution is 1920x1080 (Full HD). Although there is no formal requirement for CNN image input to ensure high accuracy, and to balance the image resolution and process time, about 20% of the full HD is set as the default size.

[Insert Figure: Data Preprocessing Pipeline]

## 3.4 Pose Estimation Model Integration

The initial plan was to develop a custom CNNs-based pose detection model to detect 6 basic human poses as output with a lightweight structure and faster process speed with minimal computation power. However, during the development, the final custom CNN model achieved over 1.2 loss on the validation dataset and only a 33% test accuracy rate. Due to the time constraint, a pre-trained model was adopted to ensure the monitor application system meets its functional requirements.

The selected YOLO11n model from Ultralytics was chosen for its balance between accuracy, computational efficiency and real-time inference capability. The model detects 17 human key points to estimate complicated pose analysis (Ultralytics, 2023). Integration was achieved using the Ultralytics Python library, which allows loading pre-trained weights and applying pose detection to static images. The model processes each frame and outputs key points, which are passed to a custom rule-based detector (RuleBasedFallDetector) to infer human posture.

[Insert Code Snippet: Model Loading and Initialization]

## 3.5 Fall Detection Logic

The fall detection logic is implemented using the RuleBasedFallDetector module. This module evaluates the spatial relationships between the 17 detected keypoints to classify poses.

* Angel Calculation: Computing angles between key points to detect abnormal postures.
* Position Analysis: Assessing the relative positions of key points to determine if the subject is fallen on the floor.
* Motion Patterns: The model monitors transitions from upright poses (standing/sitting) to prone (fall) over time to confirm a fall.

If a “fall” pose is detected and no fall alert was recently triggered, an email is sent to a designated contact, with a 10-minute cooldown to prevent spamming.

[Insert Figure: Fall Detection Rule-Based Flowchart]

## 3.6 Environmental Advisory System

Apart from the fall detection system in the monitor application system, the application also monitors environmental parameters (temperature and relative humidity) to provide advisories feedback. An OpenWeatherMap API was used to obtain environmental parameters based on the user’s geographical location (latitude and longitude). These values are then passed to a lightweight neural network defined in comfitness.py, which returns text-based health advice. The neural network was trained on labelled environmental data and achieved a validation loss as low as 0.08, ensuring reliable outputs.

[Insert Table: Environmental Parameters and Advisory Thresholds]

## 3.7 Tools and Technologies

The development utilised the following tools and libraries:

| **Tool/Library** | **Purpose** |
| --- | --- |
| **Python 3.13.1** | Core development |
| **Flask** | Running the web-based interface |
| **OpenCV** | Webcam image capture |
| **Ultralytics** | Pose estimation model (YOLO11n-pose) |
| **NumPy** | Image array manipulation and numerical processing |
| **Matplotlib** | Visualizing training loss (for CNN) |
| **smtplib & email** | Sending alert emails |
| **Threading** | Running parallel tasks for real-time frame handling |
| **OpenWeatherMap API** | Environmental data acquisition |

## 3.8 Constrains and Limitations

The monitor application system’s performance is subject to certain constraints:

* Lighting Conditions: Poor lighting can affect pose estimation accuracy.
* Occlusions: Partial obstruction of a person may prevent accurate pose estimation.
* Image Artifacts: Static images or posters showing people may falsely trigger fall alerts.
* Hardware Limitations: Performance depends on CPU capacity; slower devices may drop frames.
* Internet Dependency: API calls for weather data and email alerts require an active internet connection.

## 3.9 Summary

This chapter presented the step-by-step methodology for building the Monitor Application for High-Risk Independent Residents. It covered the rationale for selecting YOLO11n-pose over a custom CNN, the design of a rule-based fall detection engine, and the integration of an environmental advisory system. The use of lightweight, open-source tools ensures broad compatibility and ease of deployment.

# Chapter 4 – Requirements

# 

## Introduction

This chapter outlines the system requirements for the Monitor Application for High-Risk Independent Residents. These requirements were derived based on user needs, system constraints, literature analysis, and technical testing. The purpose of this application is to passively monitor vulnerable individuals, detect falls using a webcam and a pose estimation model, and provide timely alerts and environmental health advice.

The requirements are categorised into: functional requirements, which define what the system should do, non-functional requirements, which specify performance, reliability, and usability aspects, pose estimation requirements, which elaborate on the conditions and expectations related to real-time posture recognition, constraints and assumptions, which identify limitations of scope, environment, and hardware, and future requirements, which outline potential enhancements to the system.

## 4.2 Stakeholders and Use Case Context

This application primarily targets individuals at high risk of falling, such as elderly people living alone. It is designed to support their independent living by detecting dangerous events in real time.

The primary stakeholders are the high-risk individuals who may fall or experience a health emergency while alone at home most of the time. The secondary stakeholders are caregivers or relatives who need to be notified in the event of a fall. The application operates without the need for physical interaction from the user and minimises privacy concerns by avoiding recording or use of cloud-based data storage.

## 4.3 Functional Requirements

Functional requirements describe the essential operations the system must perform to fulfil its objectives:

* The system should be able to capture images from a webcam at a minimum rate of 2 frames per second (FPS).
* The system should be able to use a pre-trained pose estimation model to estimate human pose from each frame.
* The system should be able to estimate and classify the estimated poses into one of four states: standing, descending, lying, fall.
* The system should be able to prevent spamming of the recipient’s email by enforcing a minimum 10-minute delay between consecutive alerts.
* The system should be able to inform a person using email when a fall is detected.
* The system should be able to allow user to turn off the webcam whenever they wanted to.
* The system must ensure that it runs locally to prevent any potential data leakage.

## 4.4 Non-Functional Requirements

The following requirements have been selected as optional components for the Monitor Application. Implementation of them can improve the functionality of the application, although it is not mandatory and will not affect the application to be a functional software.

* The system shall process and classify each frame within 0.5s.
* The application shall require no user interaction after launch.
* Modular code for maintainability and reduce the difficulty of adding new functionality to the Monitor Application.
* The system should be able to run on Windows or Linux with standard CPU as basic requirement.
* The system should be able to remain operational for pose detection offline.
* The system should be able to retrieve temperature and relative humidity data via an API, based on user geolocation.
* The system should be able to process environmental data through a neural network and generate a text-based advisory message.

## 4.5 Future Requirements

The following requirements have not been implemented; however, they are recommended for future development to improve the Monitor Application:

* Activity tracking: This system records an individual’s daily activities with each activity’s frequency and duration. This allows the users to have a better understanding of their health level.
* Recorder and reminders for taking the pills: This is the system that can identify individuals’ actions of taking drugs and send specific pre-set reminders if no evidence of pill-taking is detected.
* Manual override panel for caregivers to confirm or cancel alerts.

## 4.6 Constraints and Assumptions

Due to the nature of the Monitor Application, and limitations in development time and available resources, this application is subject to a series of academic and technical constraints. No dedicated GPU is used for inference, the application must run on a general-purpose laptop. No real high-risk individuals were monitored during development or testing. The accuracy of pose estimation depends on sufficient ambient lighting. The webcam must be correctly positioned and functional to ensure a suitable area is covered and clear images are captured. The system assumes an active internet connection for environmental data retrieval and email.

This system is not intended for use in clinical or emergency settings.

## 4.7 Summary

This chapter defined the detailed requirements of the Monitor Application, including the necessary functional behaviours, performance expectations, hardware and environmental constraints, and future upgrade possibilities. These requirements form the basis for system design decisions covered in the following chapter.

# Chapter 5 – Design

## 5.1 Introduction

This chapter presents the detailed design of the Monitor Application for High-Risk Independent Residents. The design process was informed by the requirements established in Chapter 4 - Requirements, with an emphasis on ensuring modularity, maintainability, and real-time responsiveness.

This chapter outlines the system architecture, data flow, and major components, including the pose estimation module, fall detection logic, environmental advisory system, and user interface. The rationale behind design decisions and trade-offs is also discussed to demonstrate how technical choices support system goals and constraints.

## 5.2 System Overview

The Monitor Application is a lightweight, real-time monitoring solution developed using Python. It captures live webcam input, applies pose estimation to detect potential falls, issues alerts via email when a fall is detected, and provides health-related advice based on ambient conditions. The system is comprised of the following core components:

* Webcam Input Handler
* Rule-Based Fall Detection for Pose Estimation Module (from Ultralytics)
* Environmental Advisory Module
* Email Notification System
* Web Interface (Flask-based)

These components are designed to operate independently but are integrated to function cohesively as a single monitoring system.

[Insert Diagram: High-Level System Architecture]

## 5.3 Software Architecture

The application follows a modular architecture, with responsibilities clearly separated into functional scripts. This enhances maintainability and allows individual components to be updated or replaced without affecting the entire system.

| **Module** | **Description** |
| --- | --- |
| app.py | Main script that runs the system, captures frames, and coordinates modules |
| pose\_model.py | Wraps the YOLO11n-pose model and handles inference logic |
| RuleBasedFallDetector.py | Implements the rule-based fall classification logic |
| comfitness.py | Neural network model for generating environmental health advice |
| Index.html | Contains HTML and frontend logic for displaying results via Flask |

[Insert Figure: Software Module Interaction Diagram]

## 5.4 Data Flow Design

The system’s internal data flow begins with the capture of an image from the webcam, which is resized and pre-processed before being passed into the pose estimation module. The resulting key points are then analysed for fall risk. Concurrently, environmental data is periodically retrieved from an API and analysed for health risks and the steps are as follows:

1. Webcam captures an image frame every 0.5 seconds.
2. Image is resized to 640×640 pixels and colour-converted to RGB.
3. Frame is passed to YOLO11n-pose to obtain 17 key points.
4. Key points are passed to the fall detection engine for evaluation.
5. If a fall is confirmed, an alert email is sent.
6. Separately, weather data is fetched and passed to a neural network.
7. The advisory output is displayed on the web interface.

## 5.5 Key Component Design

### 5.5.1 Pose Estimation Module

This module uses a pre-trained YOLO11n model from the Ultralytics library. The model takes input images and resize them into 640×640 resolution and returns coordinates and confidence values for 17 body key points (e.g., nose, shoulders, hips, knees, feet) (Ultralytics, 2023). The model is running locally on a CPU, and inference is optimised to meet the 2 frames per second (FPS) performance target.

[Insert Code Snippet: Model Inference Example]

### 5.5.2 Fall Detection Logic

The fall detection engine is implemented as a rule-based system that analyses key point geometry. It evaluates factors such as the angle of the torso relative to the floor, or whether key points (e.g., shoulders and hips) appear horizontally aligned and close to ground level and pose transitions (e.g., standing → lying) monitored over multiple frames.

The detection logic maintains state by tracking pose information across frames and uses a cooldown timer to prevent email spamming. A fall is confirmed only after a consistent detection over two or more consecutive frames.

[Insert Figure: Fall Detection Logic Flowchart]

### 5.5.3 Environmental Advisory Module

The environmental advisory module uses data from the OpenWeatherMap API to get environmental parameters which include temperature (°C) and relative humidity (%). This data is passed into a trained neural network implemented in comfitness.py, which outputs a short advisory text message. Output is refreshed regularly (30 minutes was set by default) or it can be refreshed manually and displayed on the web page.

[Insert Table: Example Inputs and Advisory Outputs]

### Email Alert Module

When a fall is confirmed, the system sends an alert email to a predefined recipient using Python’s smtplib and email libraries. The message includes timestamp, pose detected. To avoid false positives and spamming, a cooldown mechanism enforces a minimum of 10 minutes between alerts.

### Web Interface

The web interface is implemented using Flask. It performs the following functions:

1. Displays the most recent image frame from cam\_sc folder
2. Return the detected pose in text
3. Updates environmental temperature, humidity, and advice
4. Allows the user to turn on and off the webcam manually
5. Ensures the webpage is only accessible locally.

[Insert Screenshot: Web Interface with Labels]

## 5.6 Design Rationale and Trade offs

### 5.6.1 Pose Estimation

The original design proposed the use of a custom CNN-based pose estimation, but the poor test accuracy (about 33%) and long training time under time constraints, which made this unsuitable for real-time deployment. A pre-trained Ultralytics YOLO11n pose estimation model was adopted due to its accuracy and real-time compatibility with CPU performance. It uses a rule-based detector method to make it easier to fine-tune without retraining and performs well with clearly defined thresholds and explainable decision logic.

[Insert Screenshot: custom cnn loss rate]

### 5.6.2 Local Execution

To ensure privacy with the guideline of Ethic’s Review and support low-resource environments. The system ensured that no video or audio files will be stored, any screenshot is deleted after being used for the model. The web service will be running locally.

### 5.6.3 Limitations Considered

Pose detection accuracy may be affected by lighting or occlusion. Occlusion is recognised as a common challenge in object detection, which can impact model performance. There is ongoing research on this complex field as Ultralytics claimed (Ultralytics 2023). The weather information for environmental parameters depends on the internet availability.

## 5.7 Summary

This chapter has presented the detailed design of the Monitor Application for High-Risk Independent Residents. It outlined the system’s modular architecture, data flow, and the design of its key components, including the pose estimation module, fall detection logic, environmental advisory system, email alert mechanism, and web interface. Design decisions were justified with respect to performance goals, system constraints, and ethical considerations, such as local execution to preserve user privacy. The choice of a pre-trained YOLO11n model and rule-based logic was driven by the need for accuracy, interpretability, and real-time performance on resource-limited hardware. Overall, the design was shaped to achieve maintainability, responsiveness, and ease of integration. The next chapter will evaluate the system’s effectiveness through implementation and testing.

# Chapter 6 – Implementation and Testing

## 6.1 Introduction

This chapter describes the implementation of the Monitor Application for High-Risk Independent Residents. It documents how Chapter 5 – Design, was translated into a working software solution using Python and relevant libraries. The implementation process focused on modularity, readability, and compatibility with local hardware constraints. Specific attention was given to real-time responsiveness, fall detection accuracy, and effective communication of environmental advice. Key challenges during the development stage and the solutions employed to address them are also discussed.

## 6.2 Development Environment

The software was developed and tested on a local machine with the following specifications:

|  |  |
| --- | --- |
| Operating System | Windows |
| Processor | CPU |
| RAM | 16GB DDR4 |
| Storage | 512GB SSD |
| Python Version | 3.13.1 |
| IDE | Visual Studio Code |
| Libraries | Ultralytics YOLO, OpenCV, Flask, NumPy, smtplib, email, requests, torch |

These specifications were sufficient to meet the project's performance requirements for image processing, neural network inference, and real-time alerting, without reliance on cloud services

## 6.3 System Setup and Folder Structure

The application was organised into a modular folder structure to enhance maintainability and clarity. Each major component had its own script or subdirectory:

* /page/app.py: Main controller that coordinates webcam input, pose estimation, and interface updates.
* /page/RuleBasedFallDetector.py: Implements the fall detection logic.
* /page/comfitness.py: Contains the environmental advisory neural network.
* /page/templates/index.html: Flask-based front-end for displaying results.
* /page/static/cam\_sc/temp.jpg: Stores the latest webcam image for interface use.

[Diagram: Folder structure]

## 6.4 Key Implementation Features

### 6.4.1 Real-Time Pose Estimation

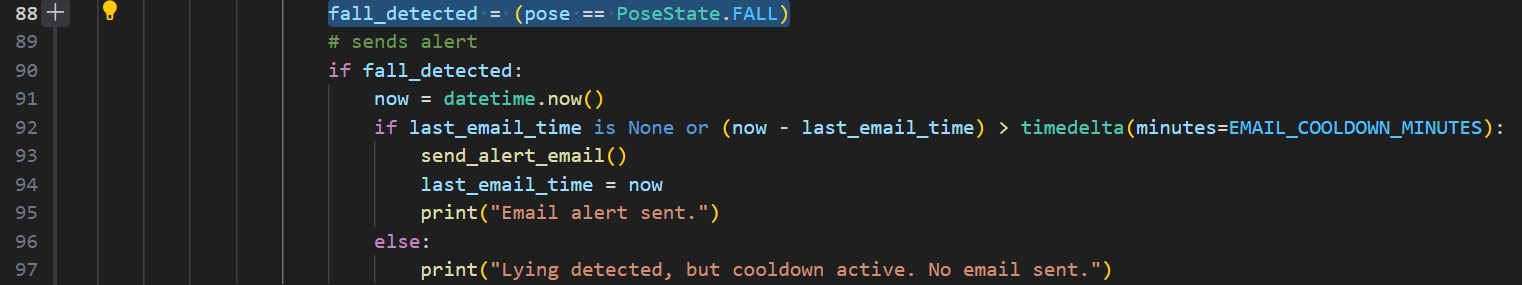
A pre-trained YOLO11n model was used due to its balance of inference speed and accuracy. The model was handled by RuleBasedFallDetector.py within the pose\_model folder and optimised for CPU execution by limiting image size and disabling augmentation during inference. The system processes two frames per second, as determined in the requirements. The YOLO11n model identifies 17 anatomical key points, which are used to reconstruct a skeletal representation for pose estimation. It returns the pose as a string for later processes.



[Insert Image: Sample Key point Overlay on Captured Image]

### 6.4.2 Fall Detection Module

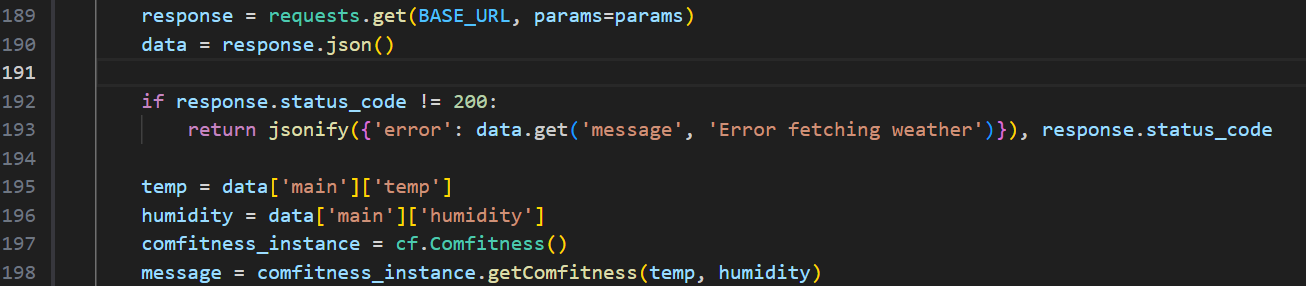
The fall detection sector was implemented as a stateful rule-based classifier that compares key points across frames to identify pose transitions indicative of a fall. Logic was encapsulated within RuleBasedFallDetector.py as described in 6.4.1 Real-Time Pose Estimation. When a fall is returned from pose estimation module, an SMTP-based email system was configured using Gmail’s SMTP server to send an email alerts, with a cooldown mechanism to prevent repeated alerts.



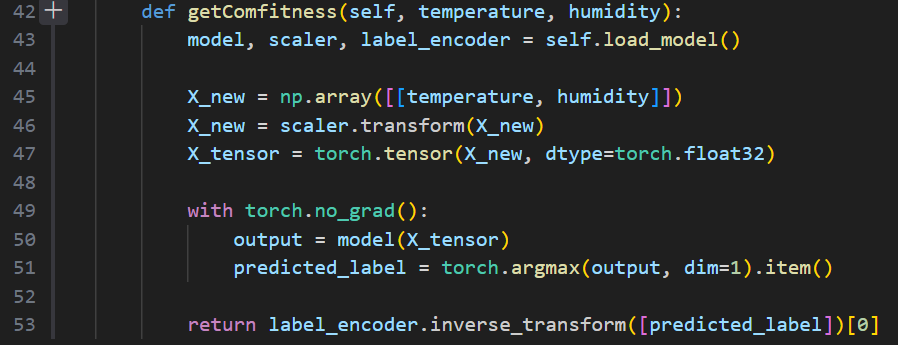
[Insert Flowchart: Fall Detection State Machine]

### 6.4.3 Environmental Advisory Module

The advisory module fetches real-time weather data (temperature and relative humidity) from the OpenWeahterMap API.



The data is passed into a lightweight feedforward neural network implemented in comfitness.py within the nn\_model folder. This lightweight feedforward neural network has 2 layers, 16 neurons in the first layer and 16 neurons in the second layer. This ensures minimal computational overhead while maintaining sufficient accuracy for real-time advisory generation. And an activation layer uses the ReLU activation function, achieved a testing loss of 0.08, and it is optimised to run concurrently with the pose estimation module on a local machine without significant resource contention. The source code for training the lightweight feedforward neural network is included in the Appendix.



[Insert Table: Example Inputs vs Advisory Output]

### 6.4.4 Web Interface

The user interface was developed using Flask and HTML. It features webcam output, environmental data, and control buttons to toggle the camera. The interface is designed to be accessible only on the local network for privacy compliance.

[Insert Screenshot: Web Interface]

## 6.5 Challenges and Solutions

* False Fall Detection:

Sudden pose transitions due to rapid movement might be misclassified as falls under low frame rate. The solution is introduced temporal smoothing and a cooldown timer to verify consistent pose transitions.

* Unstable Environmental Data:

The weather API might fail intermittently. The solution is to add exception handling and fallback logic to prevent interface crashes.

* Web Interface Rendering Delay:

Delay in image loading for refresh the webpage could cause webpage crashes. The solution is to use lightweight JPEG image format and caching strategies within the Flask route.

## 6.6 Summary

This chapter presented the implementation of each functional module in the Monitor Application, covering the pose estimation module, fall detection logic, neural network with advisory model, email system, and user interface. The implementation adhered to design principles outlined in Chapter 5 - Design, and key challenges were mitigated with appropriate solutions. The following Chapter 7 - Testing and Evaluation evaluates system performance through structured testing.

# Chapter 7 – Testing and Evaluation

## 7.1 Introduction