INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

ROBERT GORDON UNIVERSITY ABERDEEN

Multimodal Fall Detection System For Elderly Persons

Group 20 Project Proposal Document by:

Modarage Ethan Christoff Perera – 20221812 | 2331419

Senuli Laknara Wickramage – 20220950 | 2330973

Himansa Wathsiluni Jayasuriya – 20230903 | 2330903

Mevinu Induwara Gunaratne – 20232429 | 2330893

Supervised by

Mrs Vishmi Embuldeniya

Submitted in partial fulfilment of the requirements for the BEng/BSc in Artificial Intelligence and Data Science degree at the Robert Gordon University.

March 2025

Table of Contents

List of Tables

List of Figures

List Of Equations

1.0 INTRODUCTION

1.1. Chapter Overview

The aim of this project is to introduce a sophisticated fall detection system capable of predicting and detecting falls or near-fall events. Many elderly individuals, especially those over 65, are often left in isolated conditions, leading to high mortality rates due to falls and their lasting effects. Given their vulnerable state, including issues like joint aches and arthritis, many elderly people struggle with self-care.

The proposed solution is an automated system that prevents falls and ensures timely assistance when needed. By leveraging the Internet of Things (IoT), user movements and status can be effectively monitored, even in the absence of a caregiver. This document outlines the development and functionality of our fall detection system, which aims to prevent falls and alert appropriate authorities promptly.

1.2. Problem Domain

Elderly individuals aged 65 and above are particularly vulnerable to fall-related injuries, which have significantly contributed to increased mortality rates in this age group, as such injuries are often fatal. A study on a "Low-cost fall detection system" revealed that an estimated 684,000 individuals die from falls each year, with adults over 60 experiencing the highest number of fatal falls (Fitriawan, et al., 2024).

The causes of these injuries are closely tied to the isolation many elderly individuals face, with little to no supervision over their activities due to the absence or high cost of caregivers. Support services, while available, are often expensive, placing a financial burden on users. According to a study, many elderly individuals struggle to fund long-term healthcare solutions due to limited income (RIVLIN, 1988).

Even for those who can afford caregiving services, it is not feasible to ensure constant supervision. Caregivers may not always be available during critical moments or locations, as they often need to attend to other tasks. This gap in continuous monitoring leaves elderly individuals vulnerable to falls.

The increased risk of falls among elderly individuals compared to younger people can be attributed to several factors, which will be further explored in this document.

- Physical Decline due to aging as joints and muscles no longer function the same
- Chronic health problems such as arthritis can cause falls as well
- Sensory impairments such as a detachment from their sight could lead to walking into objects unexpectedly, hence causing them to fall
- Medical Side Effects may be another cause given the potency and severity of the drugs consumed by the target populi

Some of the consequences experienced by these individuals may be permanent and could quite potentially lead to a death inducing injury, so to prevent the possibility of such a process occurring a solution where the individual is closely monitored is required.

1.3. Problem Definition

Elderly individuals, particularly those over 65, face a high risk of falls due to age-related physical decline, sensory impairments, and chronic conditions like arthritis. These factors, combined with medication side effects, significantly increase their vulnerability. Unfortunately, many elderly people live alone, often without access to immediate help when they fall, leading to serious injuries or even death. Falls are a leading cause of accidental injury deaths among this age group, with long recovery times or permanent disability being common outcomes.

While caregivers provide some relief, they are costly and not always available. Family members also can’t always be present, leaving these individuals at risk. A reliable, cost-effective solution that can monitor and predict falls in real-time, such as an IoT-based system, could allow elderly individuals to live independently while reducing the risks associated with falls, improving their safety and quality of life.

1.4. Research Motivation

This research is motivated by the need to address falls among the elderly, a leading cause of injury often disregarded and overlooked in existing solutions. By developing an enhanced, cost-effective system that combines real-time fall detection, prediction, and prevention through advanced data analysis and joined models, we aim to significantly reduce fall incidents. The goal is to empower elderly individuals to live more independently and safely, providing timely interventions, minimizing injury risks, and offering peace of mind to families while improving their overall quality of life.

1.5. Existing Work

Table 1: Table for Literature Review

Citation	**Technology/Algorithm Used**	**Dataset**	**Advantages**	**Limitation**	**Metric**
Posture Detection using Image Processing API for fall detection					
(KANDAGATLA, 2022)	Makesense.ai utilized to create labels for each move made in the dataset such that each image has a label appended to it	Fall_dataset	Can be utilized to develop a complex fall detection system using the provided dataset	Due to the poor joint mappings because of poor visibility for some joints, there may be some false positives	Number of non-falls: 3124 Number of falls: 3784
Use of Accelerometers and Gyroscopes along with ML models for fall detection					
(Li, et al., 2009)	A threshold-based fall detection algorithm using tri-axial accelerometers and gyroscopes. Divides human activities into static postures and dynamic transitions.	The dataset includes activities of daily living (ADL), fall-like motions, and different types of falls (e.g., forward, backward, on stairs)	Reduces false positives and negatives. Low computational cost and real-time response.	Difficulty distinguishing between jumping into bed and falling against a wall with a seated posture.	Sensitivity: 91% Specificity: 92%
Real-Time Data Analysis for Event Prediction in Fall Detection					
(Nguyen, et al., 2024)	Non-vision-based (wearable sensors) vs Vision-based (image sequences, skeleton modeling). YOLOv3-tiny (for real-time object detection) and DeepSORT (for human tracking). AlphaPose for high-accuracy skeleton	RGB-D images and skeleton sequences captured by Kinect sensors.	High detection accuracy with YLOv3-tiny and DeepSort Preprocessing reduces false positives	RNNs used struggle with long sequences Falls lasting 400 to 1600ms require precise timing	over 99% accuracy on both standard and custom datasets for fall detection.
Monitoring Heart to Predict Fall Risk					
(Malheiros, et al., 2017)	Fall detection system that utilizes an accelerometer/gyroscope for body positioning and a	Dataset consists of patients from ward that have	Real time data analysis for active tracking and	Device consumes a lot of power and placement/quality of sensors	Body, walking and falls position were

	heart rate monitor to track heart rates	had symptoms of poor heart rates and resulting falls, source is N/A	automated alerts with body positioning may be used as references for the fall detection project	may provide inaccurate and poor feedback leading to poor results when referred to	satisfactory in 100%, 90% and 60% of the cases in a controlled environment (laboratory)

1.6. Research Gap

Existing fall detection systems typically rely on a single data stream, such as motion sensors or posture detection, which limits their ability to predict falls with high accuracy. These systems primarily focus on detecting falls after they occur and often lack the capability to foresee potential fall risks. This project aims to address this gap by integrating multiple data streams, including real-time posture detection and motion sensors (gyroscope and accelerometer) for immediate fall detection, and abnormal heart rate level monitoring to assess fall risks.

By combining these streams, the system can predict potential fall events based on factors like high or low heart rate levels, while also providing rapid detection through sensor data. Additionally, user-provided data such as BMI, age, gender, weight, and heart rate (BPM) will further enhance the accuracy of fall risk predictions. This approach creates a more comprehensive and proactive fall prevention solution. This project uniquely addresses this gap by considering multiple modes of input to achieve higher accuracy and fall prediction capabilities that existing research has not yet accomplished.

1.7. Contribution to the body of knowledge

1.7.1. Domain Contribution

The models will be trained to deliver a comprehensive, multilayered solution capable of detecting and predicting falls with greater accuracy. The domain contribution can be broken down into the following points:

1. Enhanced Accuracy – The combination of three main models - fall detection through posture analysis, sensor data (gyroscope and accelerometer), and monitoring of abnormal

heart rate levels leads to improved system accuracy. This reduces false positives and false negatives, making the system more reliable and effective in real-world use.

2. Fall Prediction and Prevention – The system offers a novel approach by predicting potential falls based on physiological data like heart rate levels. While vision data and sensor data detect falls, abnormal heart rate measurements, such as unusually high or low readings, signal the risk of a fall. This allows the system to notify caregivers or medical professionals, enabling preventive measures to be taken before a fall occurs, which adds a critical layer of early fall prediction and prevention. This aspect has not been thoroughly explored in previous systems.
3. Broader Involvement in Elderly Care Systems – Given that falls are one of the leading causes of injury among the elderly, the implementation of such a comprehensive system can have a significant impact on improving safety for this vulnerable population. By offering real-time monitoring and predictive alerts, our system empowers caregivers and healthcare providers with valuable information, enabling them to make better decisions regarding the care and well-being of elderly individuals.

Additionally, the project contributes to the field of data science by promoting the use of multimodal systems. With models that integrate posture detection, sensor data, and physiological measurements, our project supports the development of hybrid models, which remains a broad and actively researched area in data science today.

1.1.1. Technological Contribution

The key technological contribution lies in the multi-modal integration of different input streams, from posture detection to sensor data (gyroscope and accelerometer), combined with heart rate monitoring. This integration enhances the system's ability to predict and detect falls with greater accuracy.

By assessing the user’s physiological state (specifically through heart rate levels) along with physical postures and movements, our system introduces an innovative approach to fall detection. To summarize this segment, even though fall detection is a well-established field, the following points highlight our project's unique contributions to the technological domain:

- Multimodal data integration
- Accurate and immediate real time fall detection
- Hybrid Model Development

1.8. Research Challenges

- **Data Fusion and Synchronization** – Synchronizing data from multiple sources (posture detection, motion sensors, heart rate monitoring) with different sampling rates is challenging. Proper data fusion is crucial to ensure accurate real-time performance.
- **Real-Time Processing and Computational Load –** Handling multiple data streams in real time can strain system resources, particularly on mobile or wearable devices. Optimizing for speed and accuracy without overwhelming the system is a significant challenge.
- **User Variability and Adaptation –** User differences in movement patterns and heart rate responses require models that adapt to individual needs. Designing a flexible system to handle this variability adds complexity to the development.
- **False Positives and False Negatives** – Balancing sensitivity and specificity are critical to reducing false positives (incorrect fall alerts) and false negatives (missed falls), ensuring reliable and accurate fall detection.
- **Privacy and Security Concerns –** Continuous monitoring raises privacy concerns. The system must securely handle sensitive physiological and movement data while maintaining user trust.

1.9. Research Questions

1. How can real-time data from sensors and monitoring devices be effectively integrated to ensure the accuracy and timeliness of fall detection and prediction?
2. To what extent could heart rate analysis reliably predict an individual's likelihood of falling in comparison to other risk factors?
3. How will the system differentiate between fall-related movements and non-critical activities to minimize false alarms in everyday scenarios?
4. What methods will be used to assess the effectiveness of the system in a real-world setting, and how will the results be measured to ensure reliability and scalability?

1.10. Research Aim

To conclude what the research's aim is, it is to attain a system that is capable of detecting and predicting a fall that a user is to experience before they can experience it such that they are instead saved from it, this solution is also to be a more cost effective and friendly one such that it is more inexpensive when compared to regular healthcare.

1.11. Research Objectives

Table 2: Table for research objectives

Research Objective	Explanation	Learning Outcome
Problem Identification	**RO1: Development of a Multimodal Fall Detection System**: The project successfully designs and implements a multimodal fall detection system that combines data from motion sensors, posture detection, and heart rate monitoring, offering a comprehensive approach to detect and predict falls among elderly individuals. **RO2: Improved Accuracy of Fall Detection and Prediction**: The system demonstrates enhanced accuracy in both fall detection and prediction through the fusion of multiple data streams, reducing false positives and false negatives in comparison to existing fall detection systems. **RO3: Real-Time Processing and Alerts**: The system achieves efficient real-time data processing, allowing for timely alerts to caregivers or emergency services when a fall is detected or predicted, improving response times and potentially preventing serious injuries. **RO4: Integration of Heart rate Monitoring for Fall Prediction**: The inclusion of ambulatory heart rate monitoring (ABPM) successfully predicts potential falls by detecting heart rate-induced risks, adding a predictive layer to the system's capabilities. **RO5: User Adaptability and Customization**: The system is designed with adaptability in mind, allowing it to cater to individual users by learning their unique movement patterns and health conditions, leading to more personalized fall detection and prevention. **RO6: Addressing Privacy and Security Concerns**: The system ensures that user data, including motion and health metrics, is securely stored and transmitted, addressing privacy concerns associated with monitoring elderly individuals in their homes. **RO7: Cost-Effective Fall Prevention Solution**: The project demonstrates that an IoT-based fall detection system can be implemented as a cost-effective alternative to full-time caregiving services, making it accessible to a wider range of elderly individuals living independently.	LO1
Literature Review	The literature review aims to cover already explored and covered research's that have been carried out over our project, inclusive of them following similar ideas and concepts. Having listed these similar works out, we also intend on pointing out useful documents and articles that cite and prove certain facts and claims our study makes within the domain of elderly people (age 65 +) falling. Finally, to address ethical constraints, we intend on referring to papers that explain how we may tackle these issues and limitations such that they are properly addressed and dealt with, etc.	LO1
Data Gathering and analysis	• Interviews with medical professionals with the fields of physiotherapy and elder care to understand their needs and probabilities into how they are to fall • Questionnaires on how useful this may be to isolated elders to see how useful the project is to be	LO2, LO3

	• Data for research papers are to be collected from IEE Data Port, Google Scholar, etc • Journal, Articles, Books published within 2021 to 2024	
Research Design	Quasi-experimental design – since randomization is not possible due to ethical and logistical constraints, and we are comparing the outcomes of pre-existing groups to evaluate the effectiveness of different fall detection algorithms in elderly individuals. (Scribbr, 2024)	LO3, LO4
Implementation	1. The implementation of a web-based UI to handle and manage interactions with the system is to be expected 2. Implementation of a machine learning model to take in new data without relying on trained data is to be expected as well 3. Develop a real-time multimodal fall detection and prediction system by integrating posture detection, motion sensors, and monitoring data streams into a unified machine learning model.	LO2, LO3, LO4
Testing and Evaluation	Surveys and questionnaires will be used to gather feedback from end-users, caregivers, and healthcare professionals regarding the usability, effectiveness, and reliability of the fall detection system, alongside pilot testing to evaluate its performance in real-world scenarios (QuestionPro, 2024)	LO2, LO4

1.12. Project Scope

1.12.1. In scope

No	Description
1	Predict if a person is about to fall
2	Monitor the users' vitals such that they are not put at a risk
3	Alert the proper authorities when such an event
4	Monitor the user's heart rate, body position using the gyroscope and accelerometer, and posture through the camera

Table 3: In-scope project elements

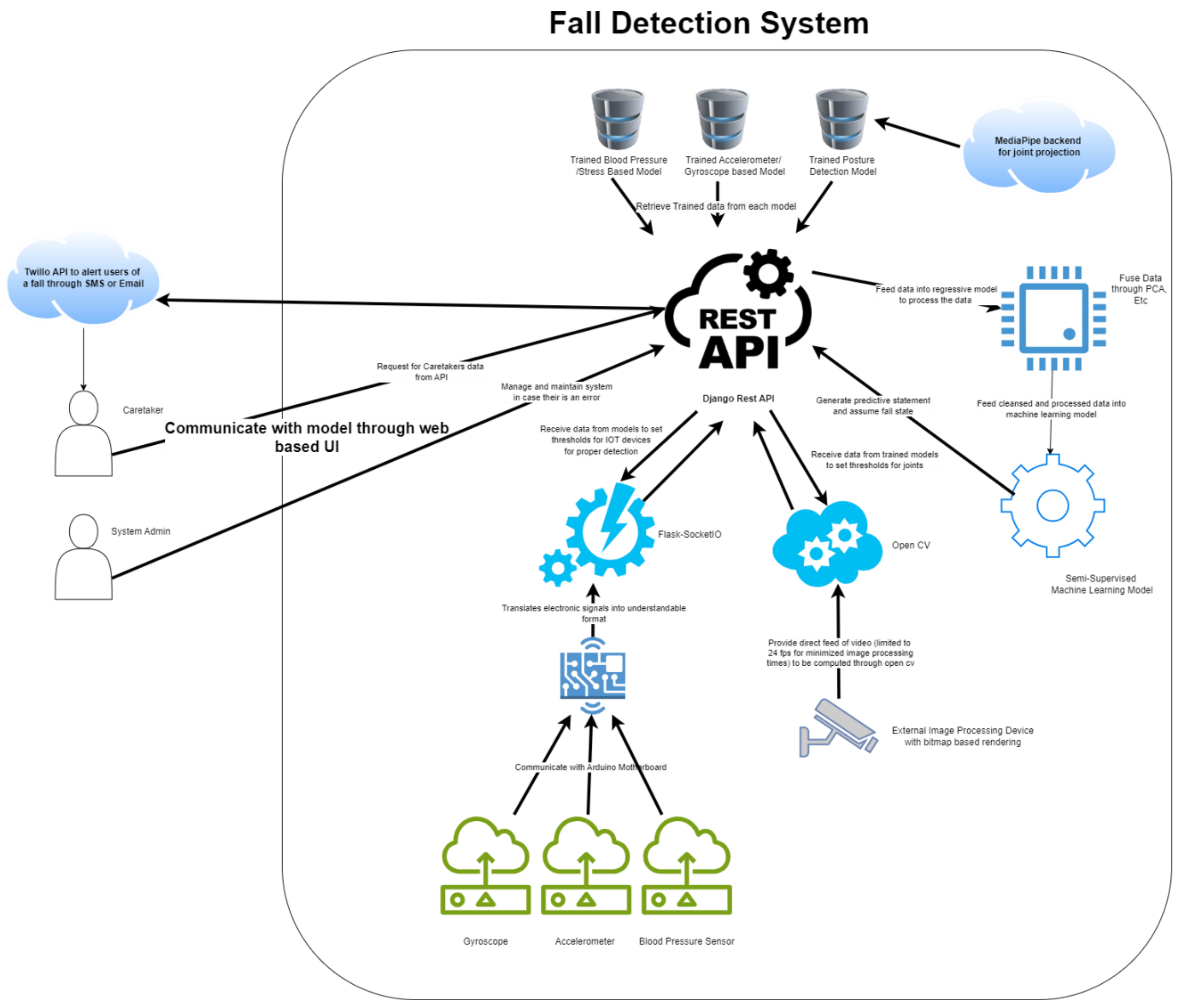
1.12.2. Out scope

No	Description
1	Make the web UI available on all kinds of devices with all kinds of language support
2	Make an application instead of a web-based UI for better performance and quality in terms of user engagement
3	Amend the model to develop medical reports of the user with the ability to pinpoint potential medical conditions as well

4	Train the model such that it can detect long term illnesses as well while making amends to the hardware such that it may be used for a longer time

Table 4: Out-scope project elements

1.12.3. Feature Prototype



Fall Detection System
Trained Blood Pressure /Stress Based Model
Trained Accelerometer/ Gyroscope based Model
Trained Posture Detection Model
MediaPipe backend for joint projection
Retrieve Trained data from each model
Twillo API to alert users of a fall through SMS or Email
REST API
Django Rest API
Feed data into regressive model to process the data
Fuse Data through PCA, Etc
Request for Caretakers data from API
Manage and maintain system in case their is an error
Caretaker
Communicate with model through web based UI
System Admin
Generate predictive statement and assume fall state
Feed cleansed and processed data into machine learning model
Receive data from models to set thresholds for IOT devices for proper detection
Receive data from trained models to set thresholds for joints
Flask-SocketIO
Open CV
Semi-Supervised Machine Learning Model
Translates electronic signals into understandable format
Provide direct feed of video (limited to 24 fps for minimized image processing times) to be computed through open cv
External Image Processing Device with bitmap based rendering
Communicate with Arduino Motherboard
Gyroscope
Accelerometer
Blood Pressure Sensor

Figure 1: Feature Prototype Design

1.12.3A Process Breakdown

In terms of the process taking place in the diagram above, regard the following steps to understand the general workflow:

1. The device is mounted onto the user and the sensors begin collecting data of the user the device is mounted onto.
2. The data is then passed through each API (Flask SocketIO, mediapipe and Open CV), each of which translates the inputted signals and images into a processable format.

- Flask-SocketIO : API used to translate communications made between IOT devices and python
- Mediapipe : Regarded for the joint projection libraries
- Open CV: Framework utilized to compute images and videos into numerical formats while supporting the use of MediaPipe based frameworks for joint projection

3. After the data is passed into Django it is then pushed through into the data pre-processing module where it is broken down into its key components
4. Django then retrieves the known data from the trained models and passes it onto the machine learning model to have the system compute whether the user is in the process of falling
5. The machine learning model is tasked with unifying and fusing all the data streams together through models such as TensorFlow to process the general outcome of a person that may fall
6. If the system returns a positive, an alert is sent to the caretaker who interacts with the system (through an API such as Twilio) to alert them of the user falling

1.13. Resource Requirements

Refer to the following to see the fundamental requirements of the project in terms of the mentioned components:

1.13.1. Hardware Requirements

- **MPU6050** – Combined Accelerometer and Gyroscope sensor for movement processing and tracking
- **Logitech BRIO Ultra HD Webcam** – Relatively cheap camera (alternates may be used as long as it’s a webcam) for posture analysis
- **Arduino UNO board** – For the sensors to interface with the application
- **CPU (Intel Core i7 10th generation processor or higher)** – A better processor may be ideal for optimal performance given its processing capabilities
- **16Gb ~ 32Gb of DDR4 RAM** – For processing heavy loads in the training process of each model
- **Storage (64Gb~128Gb) as a minimum** – To store the datasets and models in

1.13.2. Software Requirements

- **Python** – Main language used to process the entire model/dataset

- **C++** - For Arduino's component management
- **HTML, CSS, JS** – For the frontend web development used to maintain and develop the web application used to interface with the user
- **ReactJS** – To implement a more dynamic and interactive web application
- **PHP** – For the backend storage of important credentials the user may store (e.g. Age, height, etc)
- **Intelij or Pycharm & Vs Code** – Code spaces utilized to execute the code and carry out model training, etc
- **MS Word** – Used for documenting important notes and report-based files for the project
- **Obsidian** – Used for Markdown based notes for quick error annotations
- **GITHUB** – Utilized for version controlling and regulated submission for the code, etc
- **Windows Operating System (10 or greater)** – Main operating system utilized to host all the mentioned applications, etc. Optimal and simple to understand

1.13.3. Data Requirements

- The main requirements in terms of the project's dataset include the following in terms of each model:
  - Images that depict persons falling that may be used for training purposes for the image processing/pose estimation segment of the application
  - Average heart rates for adults over the age of 50 which include details such as heart rate, height and age
  - Falls detected based on erratic movements picked up from devices such as gyroscopes and accelerometers such that they may optimize the current model further
- The focus of the dataset is to be distributed amongst the mentioned features given the fact that there are three main models to the application

1.13.4. Skill Requirements

- Intuitive though processing abilities
- Time management
- Rudimentary problem-solving skills
- Report Writing
- Critical Thinking
- Fundamental knowledge of coding and version control

1.14. Chapter Summary

To summarize the chapter, all it covers is an introduction to the concept of the project such that its stakeholders, vision and purpose are listed out with an additional list of what the functional and non-functional requirements of the project are. The applications hardware, software and skill requirements are marked out as well to project what the financial costs of the project may be (in vague detail). Thereafter, the challenges experienced by the project (as well as the actions taken to handle them) are explained to convey how the project overcame some shortcomings it was to experience.

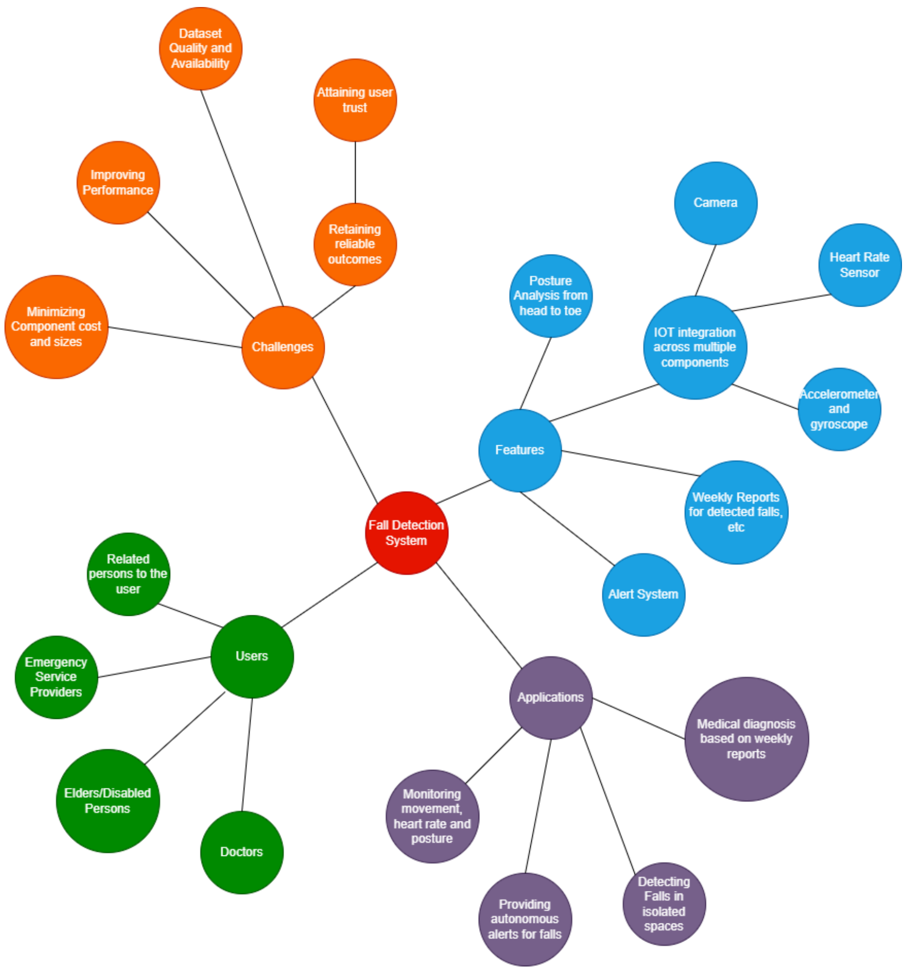
2.0 LITERATURE REVIEW

1.15. Chapter Overview

The tendency of an elderly person to fall in isolated conditions given the fact that they may be experiencing issues and illnesses with joint related issues it may be seen that there has been a vivid increase in the rate at which elderly persons tend to retain injuries at their age due to being neglected in their given state, a study was conducted from the Departments of Medicine (M.E.T., S.F.G.) and Epidemiology and Public Health (M.S.) (Tinetti, et al., 1988) and it was found that of a group of 336 elderly persons over the age of 75, 32% (108 subjects) of them had fell at least once. 24% of them had experienced serious injuries, each of which had worsened their conditions. These studies were carried out within a controlled group of people as to where each of them had no chronic disease, thus proving the fact that elderly persons with disease as such may experience much more consequential aftereffects.

Regarding the Literature review provided here, it should be noted that each of the mentioned studies have some relevance to this project in terms of how the resources they have developed and provided for their own projects might be utilized in this project. This is exclusive of utilizing the entire project but instead using small segments for each project as they each provide us with a key understanding into how we are to implement some of the components of out project.

1.16. Concept Map



Dataset Quality and Availability
Attaining user trust
Improving Performance
Retaining reliable outcomes
Minimizing Component cost and sizes
Challenges
Camera
Heart Rate Sensor
Posture Analysis from head to toe
IOT integration across multiple components
Accelerometer and gyroscope
Features
Weekly Reports for detected falls, etc
Fall Detection System
Alert System
Related persons to the user
Emergency Service Providers
Users
Applications
Medical diagnosis based on weekly reports
Elders/Disabled Persons
Doctors
Monitoring movement, heart rate and posture
Providing autonomous alerts for falls
Detecting Falls in isolated spaces

Figure 2: Concept Map

1.17. Problem Domain

The problem acknowledges elderly persons that tend to fall in isolated conditions given the fact that they are neglected, hence leading to the mentioned elderly population experiencing more falls than normal. Given the little amount of attention being given to these individuals they tend to experience falls more than normal, as a partial solution to this there are elderly care institutions that take these individuals needs and do what they can to ensure that they are treated well.

The journal articles, conferences and research papers that covers use of accelerometer and gyroscope sensors, video data-based fall detection, posture recognition and heart rate relation to falls are considered for the problem. This chapter consists of comprehensive explanations regarding these topics.

1.18. Existing Work

1.18.1. Fall Detection using Posture Detection through Image Processing

In today's world it has been found that many elderly people are affected by falls due to isolation from the presence of a more capable and caring individual. A study was conducted by a group of individuals to monitor and record the effects of social isolation on the quality of life in elderly adults (Newman-Norlund, et al., 2022). It was found that it depreciates rapidly due to the forced and induced isolation that was projected onto these individuals.

In terms of how posture detection will be used to predict a fall, it may be seen that an object detection API will first feed the algorithm with real time data that is brought in it from a device that may be staged as a camera. The first task that will take place in terms of data pre-processing will be that of how the real time data will be actively fed into the model as raw/unprocessed data. Furthermore, it may be seen that the tendency of persons over the age of 80 to fall is quite high given risk factors such as isolation that affect them (Norsk forening for epidemiologi, 2012).

Thereafter once the data has been uploaded into the machine learning algorithm it will first be cleansed and set to a limit of 24 recorded frames per second. Thereafter feeding it into the algorithm, the first process to take place is pose landmark projecting using "MediaPipe". What happens here is that the data being fed into the algorithm will first have each frame separated from one another and then processed separately. Thereafter, utilizing MediaPipe's pose landmark detection API, we will be projecting these points onto a person's detected joints.

Having projected these points onto them, the algorithm may then assume a series of thresholds that are to be surpassed to initialize a state where the person is about to fall. In other words, the algorithm will try to understand if a person's joints are surpassing a certain threshold (eg: Elbows are close to the Knees suggesting that the person may collapse), thereafter it will see if that threshold is maintained for a given period of time and then if it exceeded the necessary measure will be taken to ensure that the person falls safely while alerting the relevant authorities.

It is proposed that the peripherals utilized for this segment of the project may be cost-effective and affordable given the fact that most systems are mostly unaffordable to the necessary demographics (Eg: elderly people). So, we believe that using a trained model we may be able to implement a cost-effective solution to the issue at hand.

In terms of how we are expected to train the posture detection model, even though we propose utilizing a dataset from google we intend on customizing it with new thresholds and extended images that capture certain poses which are to be "baked" into its memory such that it may immediately identify a pose as such which is associated with a fall.

A study for the mentioned process has already been carried out by (Saraswat & Malathi, 2024) who had already implemented a vision-based fall detection system utilizing "MediaPipe" as it's backend for pose detection. To conclude, we propose that by utilizing an object detection API for posture detection we may greatly improve the overall accuracy of the system's ability to predict whether the user is about to fall or not.

1.18.2. Use of Sensors (Accelerometers & Gyroscopes) for Fall Detection

In recent years, sensor-based systems have become evident as a powerful way of detecting falls. These systems, mainly the ones that use accelerometers and gyroscopes, have the potential to continuously monitor movements and orientation. They focus on measuring motion dynamics and provide real time data to detect falls when they happen. The ability to embed these in wearable devices have enabled continuous monitoring of at-risk individuals. This chapter covers how accelerometers and gyroscopes are used in fall detection in various systems.

Accelerometers and Gyroscopes are known to be the main tools for fall detection in the past projects since they can capture data related to motion. Accelerometers are mainly used to measure the change in velocity of an object. In fall detection devices, this is used to detect falls based on sudden changes in the acceleration. In (Palmerini, et al., 2020), they have used accelerometers to detect falls based on sudden movements or impacts that might indicate a fall. Especially, if the large acceleration is followed by a still period, it triggers and alert.

When the accelerator was used alone some of the intentional motion types were also detected under falls. For example, sitting quickly. This is where the gyroscope comes in. The Gyroscopes are used to measure and maintain the orientation and angular velocity, which is most helpful in capturing data related to rotational movements of the person. A project done with the use of both sensors mentions clearly about this. (Li, et al., 2009) The combination of data from both sensors reduces the false positives that might get detected from accelerometers allows the system to distinguish between intentional motions from unintentional falls. The effectiveness of using these sensors in real world applications has been demonstrated and proved by many studies, making them valuable in elderly monitoring systems.

1.18.3. Monitoring Heart Rate Levels and Relating it to Fall Risk

Fall detection using heart rates pose a very reliable manner of detection if a person were more likely to fall or not, given the fact that if they were to experience a sudden spike in their heart rate the pressure and suddenness of it may be enough to have them collapse. Furthermore, past conditions and symptoms may regard as a further attribute to detecting if a person were to fall or not given the possibility of them experiencing similar symptoms from these past diseases again. Regarding present conditions, wearing a small device to read heart rates may help detect falls more accurately and immediately given the sensors availability and ability to provide a constant feed of real time data, not to mention the fact that this is an ideal solution for isolated and neglected individuals given its running time that may almost always go uninterrupted (Zhou, et al., 2014).

1.18.4. Real Time Data Analysis for Event Prediction in Fall Detection

Real-time data analysis is integral to modern fall detection systems, enabling immediate identification and prediction of fall events through sophisticated sensor technologies and advanced algorithms. These systems typically employ embedded sensors, including accelerometers, gyroscopes, and pressure sensors, to collect motion data from users. For instance, some systems integrate accelerometers and gyroscopes into footwear, which continuously monitor body movements (Qu, et al., 2024) This data is transmitted to a mobile application via Bluetooth, where it is processed in real time using deep learning models, such as the FallSeqTCN, designed for analyzing time-series data (Qu, et al., 2024)

In addition to using traditional motion sensors, some approaches leverage RGB-D sensors and skeleton data to detect falls (Nguyen, et al., 2024) . This method analyzes human skeleton information extracted from video footage, allowing the system to track movements accurately. Other studies enhance existing algorithms, such as the improved YOLOv8 model, which incorporates an attention mechanism to improve object localization and detection accuracy, especially in cluttered environments (Khekan, et al., 2024) .By applying these advanced techniques, these systems can more effectively distinguish between falls and other movements.

Real-time data analysis not only aids in predicting falls but also facilitates immediate alerts to caregivers or medical personnel when a fall is detected. For example, certain systems employ GSM modules to send SMS notifications and make calls, ensuring that help is promptly dispatched to the user (Fitriawan, et al., 2024). This timely response can significantly mitigate the consequences of falls, which are often more severe due to delays in assistance.

The effectiveness of these systems is evaluated using various performance metrics, such as accuracy, precision, recall, and F1 score. For instance, one model achieved an accuracy of 98% and an F1 score of 0.90, demonstrating the robust capabilities of real-time data analysis in accurately predicting fall events (Khekan, et al., 2024). These metrics reflect 7 not only the reliability of the detection systems but also the potential for scalability and further optimization as additional data and models are incorporated.

Overall, the integration of real-time data analysis in fall detection systems highlights significant advancements in ensuring the safety and independence of elderly individuals. By leveraging cutting-edge technology and sophisticated algorithms, these systems provide vital support in preventing fall-related injuries and fatalities.

1.19. Algorithmic Review

The technological approach involves a combination of sensor-based and vision-based methodologies, each with different computational techniques to improve accuracy and efficiency.

When it comes to processing the data that is collected from these sensors, there are mainly two types. Threshold based methods and more advanced machine learning based methods. A study about a comparison of these two types explains why machine learning is a better approach than the other. (Aziz, et al., 2016)

Threshold based methods lie in the earliest years, where predefined or calculated limits for acceleration and angular velocity were set and when the data exceeds them, trigger fall alert. This method resulted in a higher rate of false positives because they get triggered by non-fall activities like bending or standing up quickly.

On the other hand, machine learning algorithms like support vector machines and neural networks showed improved accuracy, due to the ability of learning patterns of real time data and hence distinguishing between everyday normal movements and fall movements.

A study (Zurbuchen, et al., 2020)has been conducted to compare the accuracy of a set of machine learning algorithms used for fall detection. They have used the dataset named 'Sisfall' (Sucerquia, et al., 2016) which is publicly available. The machine learning algorithms they have used include support vector machines (SVM), k-Nearest Neighbours (KNN), Decision Trees (DT), Random Forests (RF), and Gradient Boosting (GB). The results showed that gradient boosting outperformed other algorithms in terms of sensitivity and specificity. But it 6 also mentions the need of high computational resources and carful parameter tuning. It also mentions the importance of simple algorithms such as random forests and k-nearest neighbours due to their balance of computational needs and classification power.

Posture detection technology can be considered as an essential component of fall detection systems, since it enables real-time monitoring of body orientation to distinguish between normal activities and potential falls.

When it comes to the video-based fall detection systems they are built to utilize computer vision techniques to monitor and analyse human movements. Some often employ human pose estimation models to identify and track body key points, which then allows for detection of abnormal postures which can indicate falls.

A study by (Chen, et al., 2021) has introduced a video-based fall detection approach which leverages human pose estimation. The method has a few steps which involves extracting 2D poses from video sequences, then converting to 3D poses. A robust fall detection network that uses this approach has achieved an accuracy of 99.83% on the NTU RGB+D dataset and real-time performance of 18 frames per second on a non-GPU platform.

Another similar use of technology for video-based fall detection (Lazzi, et al., 2021) was proposed based on human posture recognition using a monocular camera. The system extracts human silhouettes from video frames. After that the data is fed to an SVM classifier which can distinguish between normal and abnormal postures. This approach has demonstrated high accuracy highlighting the effectiveness of posture recognition in identifying falls.

1.20. Tools and Techniques

The development of fall detection systems has proven to have significant benefits from integrating multiple data sources. Here it is done through sensor data, video data which will be used for posture recognition, and heart rate monitoring. The accuracy of the system in detecting falls while reducing false alarms is significantly enhanced by integrating each of these inputs. One of the main strategies that can be employed to do this is ensemble learning. By utilizing the benefits of several models, ensemble learning can increase the model's robustness. With an F-score of 94.26%, research by (Liu, et al., 2023) has shown that convolutional neural networks (CNNs) and gated recurrent units (GRUs) greatly enhance fall detection performance. (Liu, et al., 2023) also showed that convolutional neural networks (CNNs) and gated recurrent units (GRUs) greatly enhance fall detection performance.

In terms of the ensemble techniques that can be used for improving model performance they include these techniques: bagging also known as bootstrap aggregating, boosting, and stacking. These methods are especially helpful in fall detection systems that integrate data from wearable sensors, video-based posture identification, and physiological monitoring.

The different tools and equipment also have an impact on how well the fall detection systems work since that's where the data is obtained from. To identify falls, movement data is gathered using wearable sensors such as gyroscopes and accelerometers. Heart rate monitors are utilized to provide fall incidents with a physiological context. Additionally, video-based posture data was recorded using cameras. Thermal sensors and RGB-D are two examples that have been employed in earlier projects. Mobile applications that use smartphone sensors to detect falls in real time are another method used to construct this system. This was illustrated by (Wu, et al., 2019), who created a mobile-cloud collaboration system that uses smartphone sensors to detect falls in an efficient manner.

To handle and analyze fall detection data, machine learning is essential. Most fall detection systems employ Python libraries like TensorFlow, PyTorch, and OpenCV. Additionally, they provide real-time analysis of sensor and video data using deep learning models. Fall detection systems have improved the safety of those in danger by incorporating various tools and strategies to produce more precise, responsive, and scalable solutions.

1.21. Chapter Summary

To summarize what this chapter provides, is a comprehensive review of existing literature which are relevant to fall detection systems, highlighting key studies, technologies, and methodologies that help in the development of this project. The review explored the impact of fall detection among the elderly due to it being a major concern today, with significant injury rates even among those without chronical illnesses. Understanding these risks and the existing research has given the assistance and guidance for the development of an effective detection system.

Various fall detection methods were analysed in the chapter which includes posture recognition using image processing, sensor–based approaches employing accelerometers and gyroscopes, relating heart rate monitoring to falls and real time analytics. By integrating insights from existing research, this project aims to create a practical and an accurate solution which is also capable of enhancing the elderly and patient care and minimizing fall related injuries.

3.0 METHODOLOGY

3.1 Chapter Overview

In terms of what is covered in the chapter overview, the following is what is considered. The manners in which the project is to carry out research into its necessary domains, the way project is to be managed and the development of the project over time will be explained here. Projections for why certain amends need to be made to the project and other similar adjustments are explained in this chapter

3.2 Research Methodology

Research Philosophy	The author of the research has selected positivism as the research philosophy. Positivism is a research philosophy that focusses on using observable and quantifiable facts in developing knowledge. This method emphasizes testing the theories and hypotheses through data collection and analysis and then reaching object conclusions. This lines with the principles of science. In this study, the detection and prediction of falls relies on real-time sensor data. The prioritization on quantifiable data, and the results being based on measurable evidence rather than subjective interpretation make this approach adequately felicitous for this study.
Research Approach	We will adopt a deductive approach, starting with a hypothesis that factors such as posture, heart rate, motion speeds, and angular velocities can predict falls. This hypothesis will be tested through data collection and analysis from sensors and monitoring devices. The approach is suitable as it allows for testing pre-established correlations between variables and drawing conclusions based on measurable evidence.
Research Strategy	We intend on using interviews (qualitative data gathering), questionnaires and forms (quantitative data gathering) for our research.
Research Choice	Multi Method – In order to consider both the qualitative and quantitative components of the study we intend on regarding the multi-method approach as it takes into consideration the factors that require an in-depth analysis (such as ethical constraints)
Time Zone	A cross-sectional time frame will be used for this research, as it is intended to occur at a single point in time.

Table 5: Research Methodology Table

3.3 Development Methodology

In terms of the type of methodology our group will use, a "**scrum**" would be the most optimal as it utilizes an iterative and incremental agile framework (type of framework where the project is faced with iterative procedures where it goes through multiple assessments and revisions to maximize its accuracy, etc) for managing the projects development. A benefit of using Scrum as our development methodology is that it breaks the project down into smaller tasks called “**sprints**” where the workload is mitigated into small and feasible tasks that minimize time consumption and maximize productivity. Furthermore, scrum refers to the use of an **“Object Oriented Analysis and Design”** (OOAD). This is since Scrum has a modular approach to task management as it breaks down the project into smaller and more manageable tasks while maintaining incremental and iterative development processes. For the project developments life cycle our group may use a spiral model since it is appropriate given its conditions. The given PDLC is an iterative life cycle model where the project is developed in small incremental iterations where its iterations are like a sprint from a scrum methodology. Besides that, using a spiral management system would enable out group to detect and mitigate issues before they could occur such that the risk of a total failure is avoided. To conclude, the idea is scalable and compatible with an Object-Oriented Analysis and Design approach.

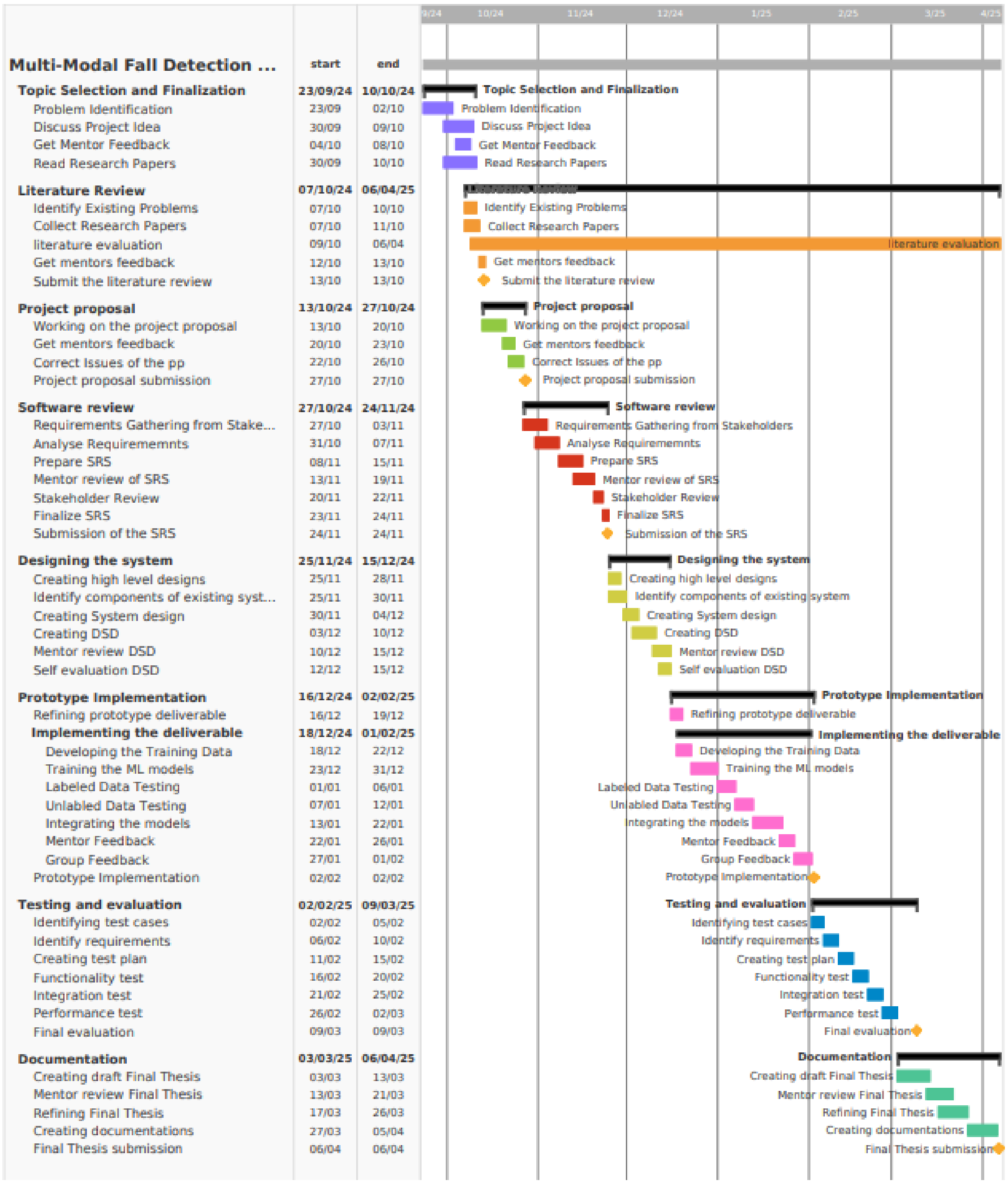
3.4 Project Management Methodology

3.4.2. Deliverables

Phase	**Deliverable**	**Week**	**Due Date**
Topic Selection	Finalized Project Topic	Week 3	10-Oct-2024
Literature Review	Literature Review Report	Week 4	13-Oct-2024
Project Proposal	Project Proposal Document	Week 6	27-Oct-2024
Software Review	Software Requirement Specification (SRS)	Week 9	24-Nov-2024
System Design	System Design Document (DSD)	Week 11	15-Dec-2024
Prototype Implementation	Functional Prototype	Week 16	02-Feb-2025
Testing & Evaluation	Testing (Identifying Test Cases)	Week 19	05-Feb-2025
Integration	Integration Process	Week 19	10-Feb-2025
CI/CD Development	Develop CI/CD Pipelines & Integrate Components	Week 21	24-Feb-2025
Final Testing	Final Testing Phase of the Applications	Week 23	09-Mar-2025
Evaluation	Evaluation Phase	Week 24	16-Mar-2025
Post-Project Analysis	Post-Mortem and Research Paper	Week 25	23-Mar-2025
Documentation	Final Thesis & Project Documentation	Week 27	06-Apr-2025

Table 6: Deliverables table

3.4.2. Schedule based on Gantt chart



Multi-Modal Fall Detection ...	start	end
Topic Selection and Finalization	23/09/24	10/10/24
Problem Identification	23/09	02/10
Discuss Project Idea	30/09	09/10
Get Mentor Feedback	04/10	08/10
Read Research Papers	30/09	10/10
Literature Review	07/10/24	06/04/25
Identify Existing Problems	07/10	10/10
Collect Research Papers	07/10	11/10
literature evaluation	09/10	06/04
Get mentors feedback	12/10	13/10
Submit the literature review	13/10	13/10
Project proposal	13/10/24	27/10/24
Working on the project proposal	13/10	20/10
Get mentors feedback	20/10	23/10
Correct Issues of the pp	22/10	26/10
Project proposal submission	27/10	27/10
Software review	27/10/24	24/11/24
Requirements Gathering from Stake...	27/10	03/11
Analyse Requirememnts	31/10	07/11
Prepare SRS	08/11	15/11
Mentor review of SRS	13/11	19/11
Stakeholder Review	20/11	22/11
Finalize SRS	23/11	24/11
Submission of the SRS	24/11	24/11
Designing the system	25/11/24	15/12/24
Creating high level designs	25/11	28/11
Identify components of existing syst...	25/11	30/11
Creating System design	30/11	04/12
Creating DSD	03/12	10/12
Mentor review DSD	10/12	15/12
Self evaluation DSD	12/12	15/12
Prototype Implementation	16/12/24	02/02/25
Refining prototype deliverable	16/12	19/12
Implementing the deliverable	18/12/24	01/02/25
Developing the Training Data	18/12	22/12
Training the ML models	23/12	31/12
Labeled Data Testing	01/01	06/01
Unlabled Data Testing	07/01	12/01
Integrating the models	13/01	22/01
Mentor Feedback	22/01	26/01
Group Feedback	27/01	01/02
Prototype Implementation	02/02	02/02
Testing and evaluation	02/02/25	09/03/25
Identifying test cases	02/02	05/02
Identify requirements	06/02	10/02
Creating test plan	11/02	15/02
Functionality test	16/02	20/02
Integration test	21/02	25/02
Performance test	26/02	02/03
Final evaluation	09/03	09/03
Documentation	03/03/25	06/04/25
Creating draft Final Thesis	03/03	13/03
Mentor review Final Thesis	13/03	21/03
Refining Final Thesis	17/03	26/03
Creating documentations	27/03	05/04
Final Thesis submission	06/04	06/04

Figure 3: Gantt Chart for project schedule

3.5 Chapter Overview

To conclude the content of this chapter, the research methodologies were covered and explained. The way each research was conducted was explained, as well as the weekly goals such that each deliverable was mentioned in the order it was to be released. Furthermore, the development methodology used to carry out the groups research was mentioned as well to point out and justify why the groups research was to be carried out that way.

4.0 SOFTWARE REQUIREMENTS SPECIFICATION

4.1 Chapter Overview

This chapter outlines the application's fundamental requirements, focusing on data collection and functional aspects influencing core functionality. It details techniques like questionnaires and interviews, discussing their procedures, pros, and cons. Use case diagrams illustrate actor interactions with the system, while stakeholders and their affiliations are identified. Lastly, functional and non-functional requirements are listed to define the application's essential needs.

4.2 Rich Picture

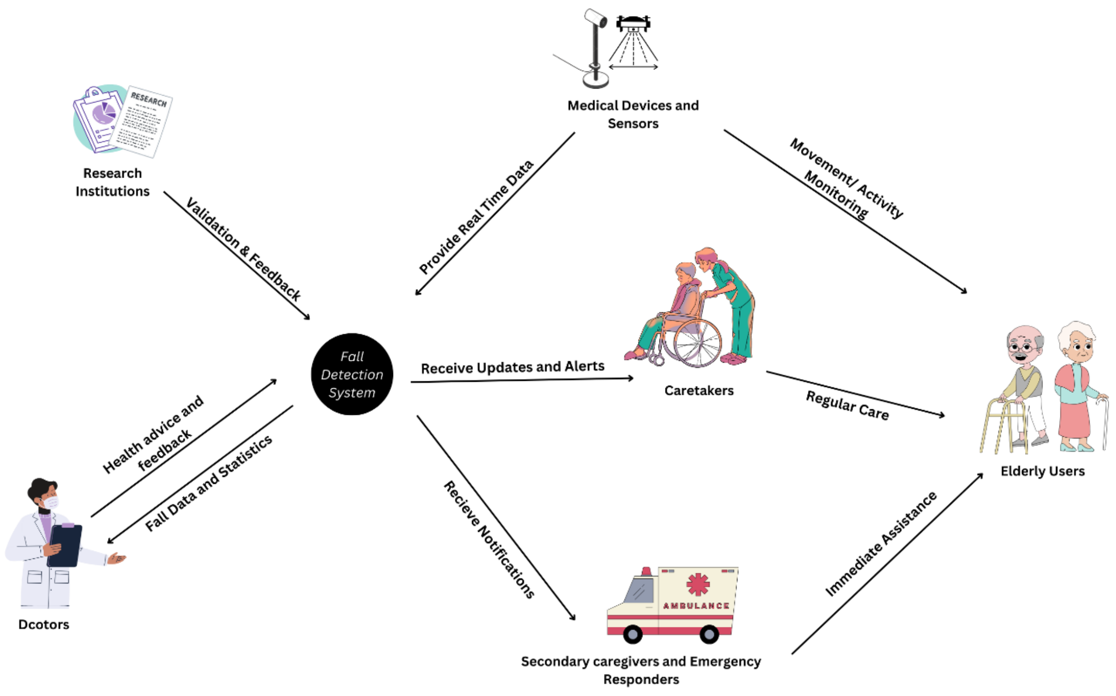


Figure 4: Rich Picture

4.3 Stakeholder Analysis

Regarding the stakeholder analysis segment of this chapter, the following Onion model lists out how a stakeholder is involved in the development of the system:

4.3.1 Onion Model

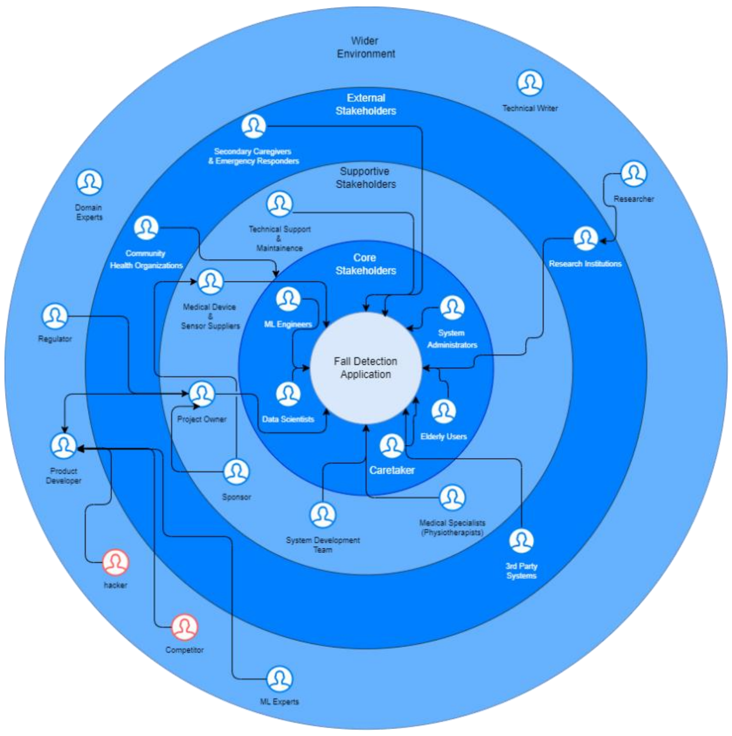


Figure 5: Onion Model

4.3.2 Onion Model Content

Stakeholder	Role in system	Contribution/ Benefit
ML Engineers, Data Scientists	Operational/ Functional maintenance	Design the process in developing a system to analyse potential falls while maintaining its functionality/accuracy and providing newly found data into data pools utilized within the machine learning model for better accuracy
System Administrator	Operational Administration	They assist with deploying the application and configuring it to fit in different environments
Caretaker	Support Provider	They provide insights into how to look after elders while taking care of them while the system is being implemented
Elderly Users	Main User	They act as the main user of the system while providing it with new training data (actively)
Technical Support & Maintenance	Project/ Application Functional maintenance	They help maintain the system if ever it were to fail while managing false positives and functional failures
Medical Device & Sensor Suppliers	Peripherals Provider	They provide the system with affordable options for the peripherals utilized in the system
Project Owner	Functional Beneficiary	Owner of the fall detection system

Sponsor	Functional Beneficiary	Fund the project through sponsors such that it can be developed further
System Development Team	Project/ Application Developers	Further develops the application to negate false positives while increasing the overall accuracy
Medical (Physiotherapist)	Health Advisors	Provide the system with advice over how to better detect falls and what parameters are to be considered in doing so
Secondary Caregivers & Emergency Responders	Medical Support Advisors/ Responders	Respond to emergencies in case a fall were to occur
Community Health Organizations	Ethical/Medical Constraint Advisors/ Responders	Regard ethical constraints the application may occur in terms of privacy and help administer solutions to it while constantly updating the systems database of constraints, etc
Research Institutions	Knowledge Contributors	Validate the accuracy of the system by carrying out recursive tests and provide evidence-based insights into the systems designs, aligning it more with real world needs
3rd Party Systems	Functional beneficiaries	Enhance system functionality by allowing it to connect with other platforms (e.g., emergency services, health records) for seamless data exchange and quick response.
Technical Writers	Operational Support	Support usability by developing clear, accessible documentation for stakeholders, including users, caregivers, and developers
Researcher	Knowledge Contributor	Improve accuracy by validating the system's algorithms, analysing effectiveness, and providing evidence-based insights for iterative improvements, they further the projects scope as well to experiment with new functions
ML Experts, Domain Expert	Expert	Enhance fall detection accuracy through improved algorithms, reducing false positives/negatives and optimizing system performance while ensuring the application caters to real world needs in actual healthcare scenarios
Competitor	Negative Stakeholder	Develops an application that directly contrasts our application
Hacker	Negative Stakeholder	Finds vulnerabilities within the system, accesses them and then reports them to the system project owner such that they are notified of a breach of privacy/functionality
Product Developer	Developer, Operational Maintenance	Ensure the system is user-centred, incorporating practical features and achieving alignment with stakeholder needs and compliance standards.
Regulator	Quality Regulator	Ensures that the application stays within healthcare-based application standards while improving patient safety, data

		privacy, and legal compliance. This boosts system credibility and user trust

Table 7: Onion Model Stakeholder Descriptor

4.4 Selection of Requirement Elicitation Techniques/Methods

4.4.1 Analysis of Requirement Elicitation Methodologies

Requirement elicitation involves various methods to define a system’s needs. This section evaluates the advantages and disadvantages of common approaches and outlines the techniques applied to develop our fall detection system.

4.4.1.1 Observing Existing Systems and Literature Review

Studying existing systems and reviewing literature provides insights into current fall detection technologies, highlighting areas for improvement.

Advantages	Disadvantages
Provides a foundational understanding of fall detection systems.	Reviewing research papers and solutions can be complex.
Helps identify feature gaps and focus on patient safety.	May lack real-world data for targeted demographics or use cases.

Table 8: Advantages & Disadvantages of 1.4.1,1

4.4.1.2 Surveys & Questionnaires

Effective for gathering input from elderly patients, caregivers, and potential users, providing a broad understanding of user needs.

Advantages	Disadvantages
Reaches a wide audience and captures diverse insights.	Responses may vary in quality or misunderstand questions.
Time-efficient and straightforward for analysis.	Limited to predefined questions, missing nuanced experiences.

Table 9: Advantages & Disadvantages of 1.4.1.2

4.4.1.3 Interviews

Interviews with medical professionals, caregivers, and patients provide detailed insights into critical requirements and user perspectives.

Advantages	Disadvantages
Allows for detailed follow-up and clarification.	Time-intensive and limits the number of respondents.
Offers unique qualitative insights.	Some interviewees may struggle to articulate requirements.

Table 10: Advantages and Disadvantages of 1.4.1.3

4.4.2 Requirement Gathering Methods Selected

A combination of methods ensures comprehensive, accurate requirements:

- **Structured Interviews**: Gather expert insights from medical professionals on fall risks and conditions.
- **Closed-Question Questionnaires**: Collect feedback from elderly patients on fall experiences and needs.
- **Unstructured Interviews**: Capture caregiving experiences and real-world challenges from caregivers.
- **Structured Questionnaires in Physiotherapy Clinics**: Collect consistent data on patients' fall risks and histories.

This balanced approach integrates qualitative insights and quantifiable data, enabling the development of a user-centred, reliable fall detection system.

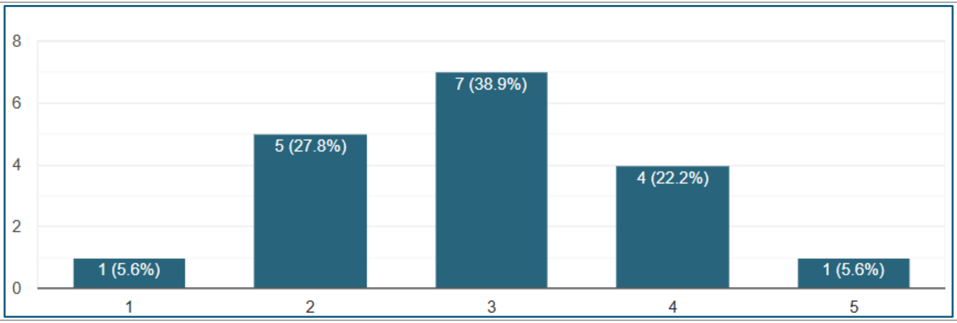
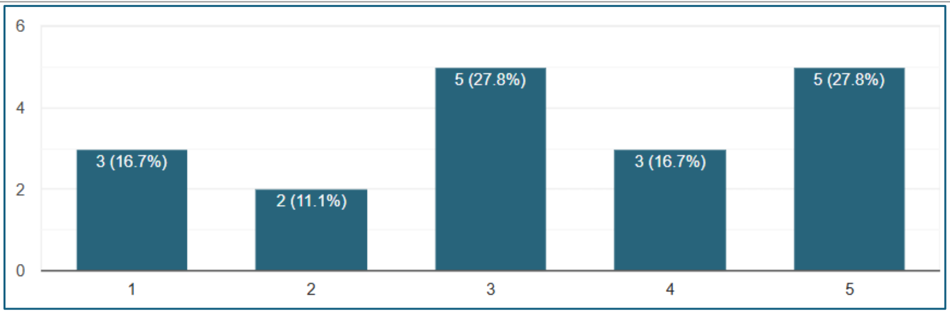
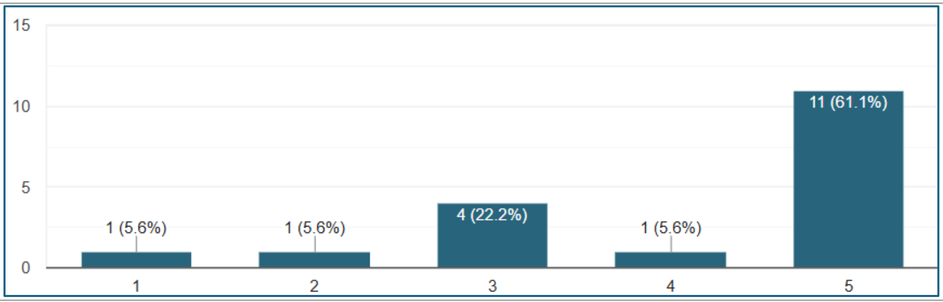
4.5. Discussion of Results

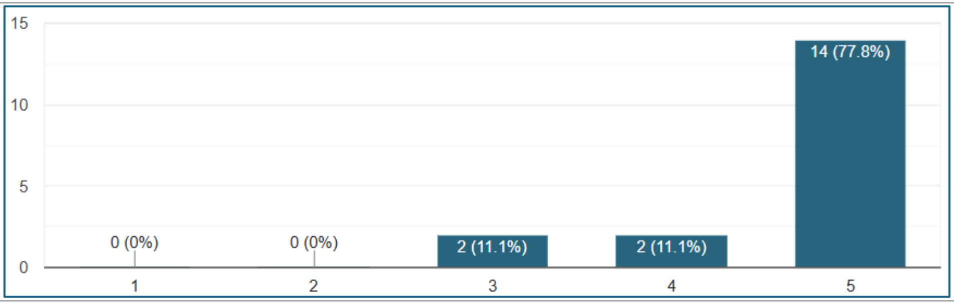
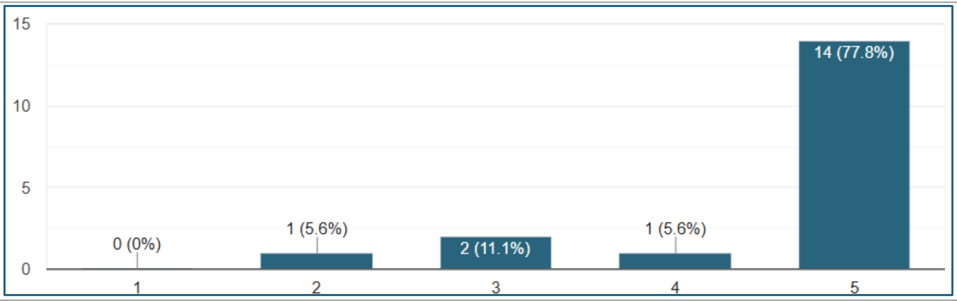
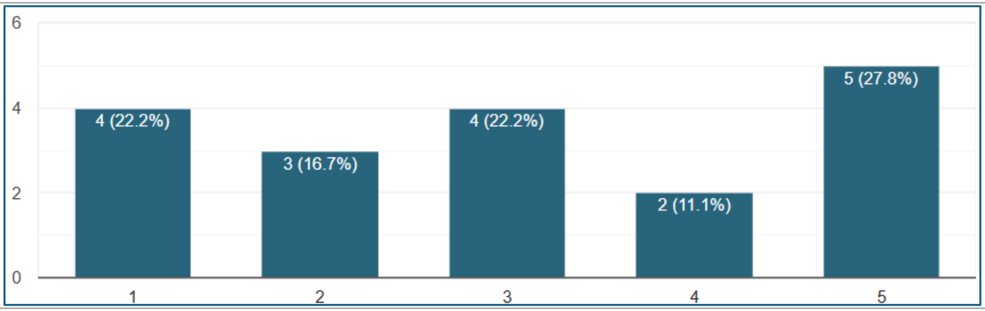
Table 11: Interview Results with the Physiotherapist at National Hospital Sri Lanka

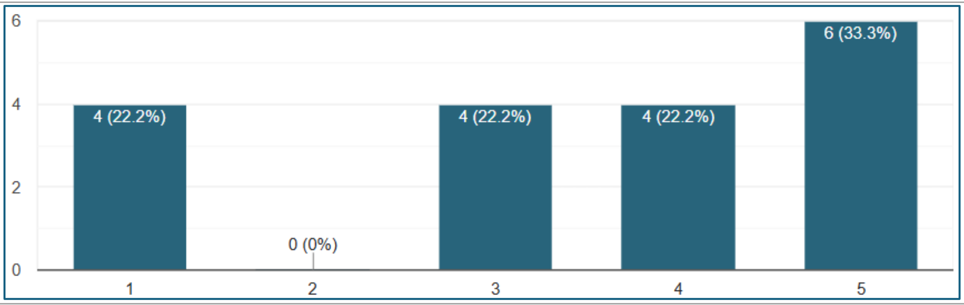
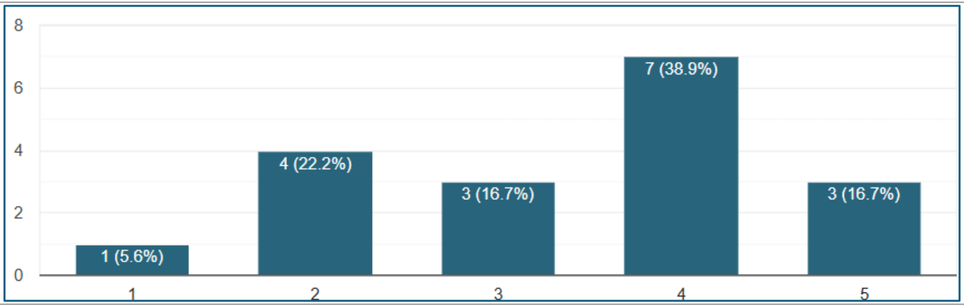
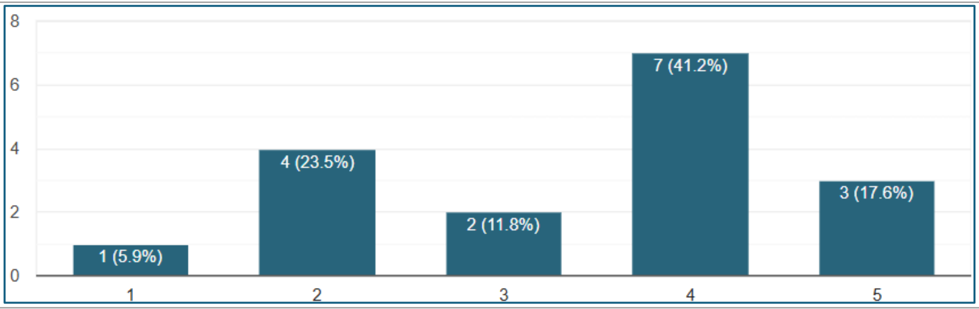
Question	**Question Aim**	**Finding**	**Conclusion**
1.Where on body is a good location to fix an accelerometer and gyroscope for accurate fall detection?	Determine the optimal location for device placement for reliable fall detection.	It was found that placing the device on the stomach is more fitting given the fact that the centre of gravity is normally located there and easily traceable	Placing the device on the stomach is ideal for accurate fall detection.
2.What health conditions or diseases might make a patient more vulnerable to falls?	Identify health conditions that increase fall risk.	Necrosis, Vascular disease, sudden drops and rises of blood pressure, loss of body mass, postural hypertension, joint issues, arthritis, neurological diseases, skeletal deformities.	Understanding these conditions helps target at-risk patients for fall prevention.
3.How quickly should the alert be sent, and should this response time vary based on individual patient conditions or be a standard time frame?	Establish an optimal alert response time to prevent falls.	In general, make it such that the application detects that a person is in the projected process of falling at least 2~3 minutes before they fall such that they may be told to take a seat.	Implementing a pre-fall alert system may effectively reduce the risk of injuries

4.If a patient receives an alert that they are about to fall, can they typically stabilize themselves in response to this alert?	Assess whether patients can respond to pre-fall alerts effectively	In imminent falls, patients can't stabilize themselves easily, but they can be notified to take preventive actions such as sitting down or getting low to the ground immediately.	Alerts can enable patients to take safer positions, reducing injury risk.
5.What assessments or tests do you use to evaluate a person's risk of falling?	Identify reliable fall risk assessment tools.	Physiotherapists calculate the risk of falling using fall risk scales - Berg Balance Scale, the Timed Up and Go test, and the 10-Minute Walk Test	These provide a standardized measure of fall risk for clinical use.
6.Can blood pressure be considered as a good factor for finding fall risk?	Evaluate blood pressure changes as indicators of fall risk.	Blood pressure is not a key factor for falls, but in sudden changes (drops or rises) falls can occur due to dizziness, loss of balance or fainting.	Looking for sudden blood pressure changes should be done rather than just high or low.
7.What additional health conditions contributes to falls, such as sudden drops in blood pressure?	Identify other conditions that may lead to falls.	Conditions like high blood pressure, low blood sugar levels, and heart attacks can also lead to falls.	Blood sugar and heart attack monitoring is not possible due to resource limitations
8.What privacy concerns should we consider, and from whom should we obtain consent?	Ensure compliance with privacy and ethical requirements	Patients consent should always be taken. The patient must always be aware of the device.	Following privacy protocols protects patients' rights and trust in the system.
9.In terms of placing a blood pressure monitor on the body, where can be an appropriate place?	Determine the optimal location for monitoring blood pressure.	Wrist is a good position, because the skin in that area is thin and arteries and veins going in that area is easily identifiable from outside.	Wrist placement should be done for accurate blood pressure monitoring.
10.How do you think our system could benefit patients, especially older adults?	Evaluate potential patient benefits of the system.	very good product for elders and patients with risk of fall. Falls cause injuries, bone fractures, get bedridden, head injuries that will lead to death. This can minimize most of them if alerted before falling.	Early fall alerts may reduce recovery time and improve patient outcomes.
11.Do you have any suggestions for improving our system?	Get suggestions to improve fall detection effectiveness.	Consider identifying elderly patients with specific conditions and directing them toward physiotherapy support.	Personalized recommendations improve system's effectiveness for at-risk patients.

Table 12: Results from the elderly individuals/patients from the National Hospital

Understanding their Needs and Experiences	
Question 1	**How often do you feel unsteady or at risk of losing balance during your daily activities?**
Question Aim	To assess the individual's perceived risk of falls during their daily routine.
Observation	 Responses mostly range from 2 to 4, indicating a moderate sense of unsteadiness.
Conclusion	Many respondents occasionally feel unsteady, suggesting some awareness of fall risks but not extreme concern.
Question 2	**How much extra support do you feel you need for specific activities?**
Question Aim	To understand the level of assistance the individual requires for various tasks.
Observation	 Ratings are mixed, with some participants rating it low (1-2), while others lean toward moderate or high (4-5).
Conclusion	The need for additional support varies widely, reflecting individual differences in physical capabilities and confidence.
Gauging Expectations for the System	
Question 3	**How important would it be to you to receive an alert if you were at risk of a fall?**
Question Aim	To evaluate the importance of a fall detection and alert system to the individual.
Observation	 Most responses are rated 5, highlighting high importance.
Conclusion	The majority consider fall alerts crucial, reinforcing the system's relevance.

Question 4	**How comfortable would you feel if someone (like a family member or neighbour) were notified in case of a fall?**
Question Aim	To determine the individual's openness to external intervention during a fall incident.
Observation	 Responses show strong agreement (mostly 5).
Conclusion	Participants are highly comfortable with notifying others, emphasizing the importance of involving caregivers or family.
Question 5	**If you couldn't reach someone right away in an emergency, how useful would you find a system to automatically request help?**
Question Aim	To assess the perceived utility of an automated emergency response system.
Observation	 Almost all responses rate this at 5.
Conclusion	There is strong support for automated emergency assistance, validating the system's utility.
Understanding Comfort Levels with Technology	
Question 6	**How comfortable are you with using technology devices?**
Question Aim	To gauge the individual's familiarity and ease with technology usage.
Observation	 Responses range from 1 to 5, with a mix of very low and very high comfort levels.
Conclusion	There is a divide in technology comfort, suggesting the need for user-friendly designs and possible training for some users.
Question 7	**How concerned are you about privacy when using a monitoring device?**
Question Aim	To understand privacy concerns related to monitoring systems.

Observation		Ratings vary between 1 and 5, with a tendency toward moderate concern (3-4).
Conclusion	Privacy concerns are significant for some participants, requiring attention to data security and transparency.	
Understanding Comfort Levels with Wearable devices		
Question 8	**How comfortable would you be with wearing a lightweight device every day?**	
Question Aim	To evaluate the willingness of the individual to wear a device consistently.	
Observation		Responses are varied, with a lean toward moderate to high comfort (3-5).
Conclusion	While most respondents are open to wearing devices, there may still be some resistance or need for adaptation.	
Question 9	**How frequently would you be willing to wear a device during the day?**	
Question Aim	To understand the individual's preference for device usage frequency.	
Observation		Most responses range from 2 to 5, showing varied preferences.
Conclusion	Most respondents are willing to use the device frequently, but some might only use it part-time.	

4.6 Summary of Findings

Table 13: Summary of findings

Findings	**Literature Review**	**Questionnaire**	**Interviews**
Optimal device placement on the body	X		X
Health conditions affecting fall risk	X		X
Importance of pre-fall alert systems		X	X

Privacy concerns in monitoring systems	X	X	X
Variability in fall assessment tools	X		
Lack of focus on multi-factor detection	X		

4.7 Context Diagram

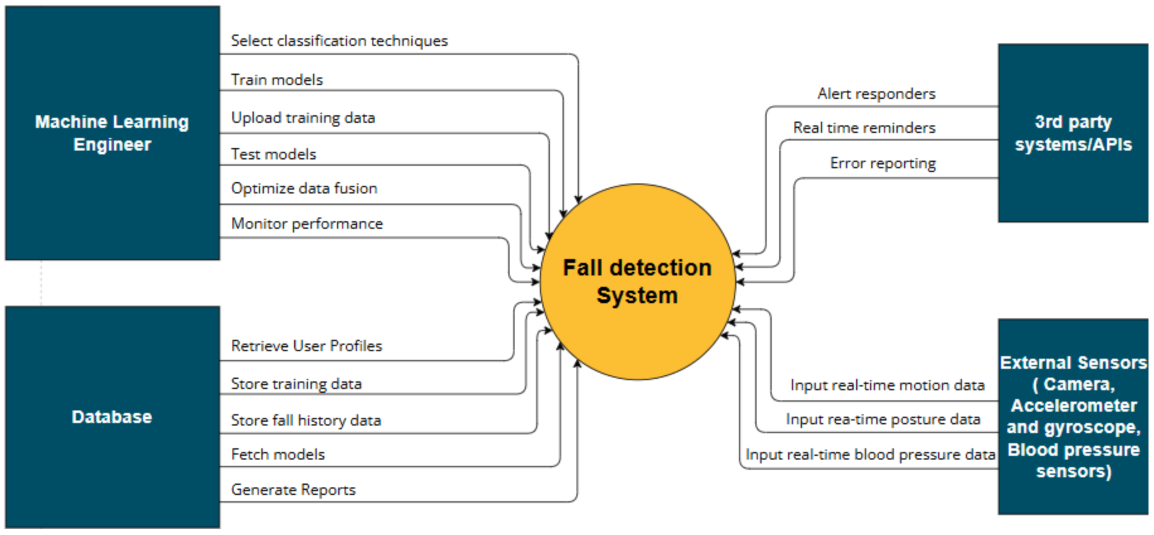


Figure 6: Context Diagram

4.8 Use Case Diagram

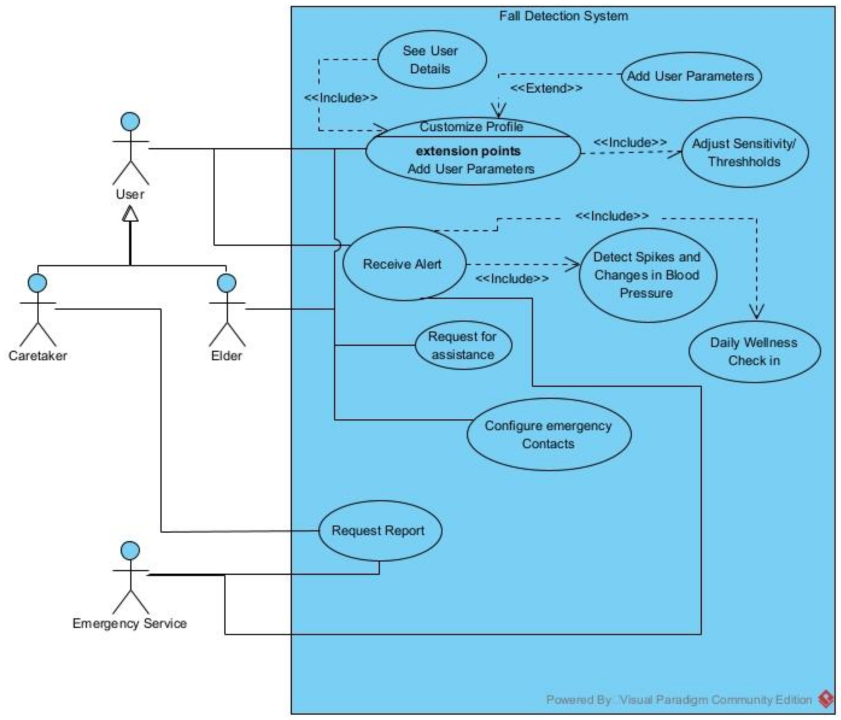


Figure 7: Use Case Diagram

Use Case ID	UC01
Use Case Name	Customize Profile
Description	Customize and view user profiles, registering details and adjusting thresholds
Actors	User (Caretaker, Elder)
Pre-conditions	User must log into the system with valid credentials
Main-Flow	1.User Customizes Profile 2.User adjusts the thresholds of certain parameters (eg: age, height, BMI 3.Add new parameters such as bone density, past conditions, etc 4.User profile is then stored in the system 5.User can view profile to identify causes of symptoms/behaviour.
Alternative Flow	-
Exceptional Flows	User fails to login, loop back to the start of UC01

Table 14: UC01 Description

Use Case ID	UC02
Use Case Name	Receive Alert
Description	User receives fall alerts (caretaker, emergency service, or elder alert to rest/exercise)
Actors	User (Caretaker, Elder), “Emergency Service”
Pre-conditions	A user profile must exist (UC01)
Main-Flow	1.An elderly individual is in a risk of falling 2.System alerts for erratic movement and blood pressure spikes. 3.Elderly individual performs exercises for self-assessment. 4.The relevant authorities are alerted in case they are in risk of a fall
Alternative Flow	Instead of receiving an alert for a negative factor, it may just be to remind the user to carry out their “daily wellness checks in”
Exceptional Flows	The system fails to alert the user so proceeds to contact another individual

Table 15: UC02 Description

Use Case ID	UC03
Use Case Name	Request for assistance
Description	Elder may press the emergency button to alert authorities.
Actors	Elder
Pre-conditions	A user profile must exist (UC01)
Main-Flow	1.The elder experiences some difficulties in walking 2.System device contacts authorities as a precaution.
Alternative Flow	-
Exceptional Flows	In case the system doesn’t work through a manual activation, the systems automated facilities will alert the relevant authorities

Table 16: UC03 Description

Use Case ID	UC04
Use Case Name	Configure emergency contacts
Description	The user can set up emergency contacts in the system.
Actors	User (Caretaker, Elder)
Pre-conditions	A user profile must exist (UC01)
Main-Flow	**1. System** shows the "Emergency Contacts" setup screen in settings. **2.User** selects the option to add a new emergency contact. **3.System** prompts the user to enter contact details (e.g., name, phone number, relationship). **4.User** enters the emergency contact information and confirms the entry. **5.System** verifies the format of the contact information (e.g., valid phone number). **6.System** saves the new contact and displays a confirmation message. **7.User** repeats steps 2-6 to add additional emergency contacts if needed.
Alternative Flow	The user could modify an existing contact or remove one
Exceptional Flows	Invalid contact information, database connection error, maximum contacts reached

Table 17: UC04 Description

Use Case ID	UC05
Use Case Name	Request Report
Description	Elevated users can request performance reports (weekly, bi-weekly, etc.).
Actors	"Emergency Service", Caretaker
Pre-conditions	-
Main-Flow	**1.Actor** navigates to the "Reports" section within the system. **2.System** displays available report options (e.g., weekly, bi-weekly, monthly). **3.Actor** selects a desired report type and specifies a time period. **4.System** retrieves data for the selected time frame and compiles the report. **5.System** generates the report and displays it to the actor in a viewable format. **6.Actor** reviews the report and may choose to download or print it.
Alternative Flow	Request a custom time period-based report
Exceptional Flows	No data available for selected period, Report generation timed out due to error, insufficient access rights

Table 18: UC05 Description

4.9 Functional Requirements

The functional requirements of the application are listed down below (they cover the fundamental needs of our application for it to be considered as a useable system):

Priority Level	Description
Critical	Core functionality/feature of the application, cannot function without it
Moderate	Not mandatory, but is considered as a requirement

Non-Important	Out of scope requirements

Table 19: Priority Definitions

ID	Requirement And Description	Priority
FR01	**Real-Time-Fall Detection** – Always analysing the potential of a fall using the sensors and the camera accurately.	Critical
FR02	**Automated Alerts before a fall occurs and after -** The system must be able to alert the relevant authorities if a user is in risk of a fall or has fallen such that they will respond immediately to the issue	Critical
FR03	**Posture Detection -** The application should be able to detect the posture of a person to detect the persons pre-fall poses.	Critical
FR04	**Blood Pressure Monitoring -** The system must be able to accurately and actively measure the users blood pressure to know if the user might experience a loss or gain in blood pressure	Critical
FR05	**Activity Monitoring -** Track the daily movement of the user to actively being traced and monitored by the system (e.g. see if they're standing, walking or in an idle position)	Moderate
FR06	**Periodic Health Check Reminders -** Constantly or time to time reminding the user to take self-assessments for them and the caretakers to understand their health status. Besides that, the user may be instructed to take a seat to keep them safe.	Critical
FR07	**Long-Term-Data Storage -** Storing historical health data (posture, blood pressure, fall events) to track trends and to generate insights over time.	Moderate
FR08	**Emergency Contact Setup -** The system should enable the user to setup emergency contacts (such as authorities in areas close to them) for them to respond to a fall or potential fall immediately	Critical
FR09	**Battery Level Alerts -** The system should be able to alert the user of low battery levels so in those cases, user can be advised to stay in an idle position (in a controlled space) until the device is ready to be used again.	Critical
FR10	**User Profile Customization -** The caretakers should be able to develop profiles for the user (elder) that take into consideration their age, height, weight, previous conditions.	Moderate
FR11	**Environmental Monitoring Integration -** The system may optionally amend the sensors for room temperature or humidity to provide a safer living environment for users in isolated settings	Non-Important
FR12	**Weekly Health Reports -** The system should be able to generate reports on the users' movements for the week to specify if they have been moving around in an irrational manner such that it might suggest that the user is experiencing fall inducing symptoms	Moderate
FR13	**Manual Emergency Button -** Provides a button for users to manually trigger an alert if they feel at risk of falling or are experiencing a health issue.	Critical
FR14	**Educational Content on Join health -** Includes access to information and tips on joint health, fall prevention, and exercises for improving stability and balance.	Non-Important

Table 20: Functional Requirements Table

4.10 Non-Functional Requirements

The non-functional requirements establish the quality and operational standards for the fall detection system, emphasizing reliability, real-time performance, and data security to ensure it effectively supports its critical functions.

ID	Requirement and Description	Priority
NFR01	**Reliability and Responsiveness**: The system must accurately detect falls and give alerts with minimum of false positives or negatives immediately.	Critical
NFR02	**Performance and Efficiency**: The immediate detection and sending alerts should be done with optimized usage of power for wearable devices.	Critical
NFR03	**Usability and Accessibility**: The interface has to be user-friendly. Needs to have accessible designs (visually impaired people)	High
NFR04	**Security and Data Privacy**: Sensitive data must be handled with privacy standards, which limits access to authorized personnel.	Critical
NFR05	**Scalability and Interoperability**: System should be able to support additional users, sensors, and devices without the need of significant changes.	Moderate
NFR06	**Maintainability**: The codebase should be able to be easily maintained, by allowing for easy updates, bug fixes and future enhancements.	High

Table 21: Non-Functional Requirements Table

4.11 Chapter Summary

To summarize, this chapter covered the fundamental and core requirements of the fall detection project as it listed out all the findings of the study, the functional/non-functional requirements, the use cases, requirements elicitation techniques and the stakeholder analysis to cover the potential contributors to the system (and its development).

5.0 SOCIAL, LEGAL, ETHICAL AND PROFESSIONAL ISSUES

5.1 Chapter Overview

Regarding the SLEP analysis of the project, it covers all the ethical and professional constraints the project is to experience in terms of the environments it is to face and be implemented in. This considers the realistic and holistic issues the project is to come across such as privacy invasions, e.t.c. Considering these probable issues, the fall detection project is to experience, a in depth analysis was carried out into order to regard the potential constraints and resolve them in an ethical manner such that they may be regarded in a professional and well documented manner, hence resulting in the following management functions.

5.2 SLEP Issues and Mitigation

5.2.1 Social Issues

- Regarding the privacy of the application, it refers to not breaching the user's privacy such that they may not reveal personal details or features (e.g. not placing the IOT device for the project in a personal space such as a washroom). Furthermore, the data collected from each user will not be redistributed to third parties
- Referring to the accessibility of the application, it will be developed in an English medium on an easy and quick to load website that consists of a simple layout that may be accessed by both the user and related clients
- To maintain access to those with cultural diversity in terms of language barriers, a Sinhala medium is to be provided to cater to those that do not understand English

5.2.2 Legal Issues

- Referring to some of the data protection laws to be instanced in the application, the following are to both be regarded and applied:
    - General Data Protection Regulation (GDPR) (European Union)– Ensures that explicit content is to be taken from the user before ever disclosing the collected data while collecting only necessary information (Intersoft Consulting , 2024)
    - Health Insurance Portability and Accountability Act (HIPAA) – Ensures that the collected health information is never to be disclosed without the patients consent to only required individuals such as a registered medical worker (CDC, 2025)
    - Personal Data Protection Act (PDPA) – Ensures that the client being monitored gives explicit consent to releasing the data for medical purposes before doing so such that all the required purposes are stated before implementing the devices and applications onto the patient (PARLIAMENT OF THE DEMOCRATIC, 2022)

- The property and ownership respectfully go to the creators and publishers of the fall detection project such that it may not be redistributed or reproduced without proper consultation
- The persons liable for the project are to be passed onto those that implement it and setup the tools to monitor the necessary features such that it has only partial affiliation to the creators of the project

5.2.3 Ethical Issues

- The user's identity is to be always kept confidential and private unless explicitly stated by the patient that it be disclosed
- The projects statements may not be 100% accurate all the time so third-party opinions are to be more fairly regarded than the applications produced results
- The user's consent is to always be regarded while being absolute such that if they wish not to disclose the collected information, it will not be released
- Privacy is to be always maintained in terms of the users more confidential data that reveal explicit details of them (such as disclosing personal/private details from especially the camera feeds)
- Only restricted individuals are to retain access to the application such that only they may have access to the patient's data

5.2.4 Professional Issues

- GitHub and Jira are to be used for version control and issue management/code reviews while retaining team-based collaboration and workflow management
- Whatsapp, Email and teams are to be used for official communication within the projects team and other related individuals
- Supervisor meetings are to be held once to twice a week through google meets
- Official Communications are to only ever be conveyed through emails

5.3 Chapter Summary

What this chapter covers are the fundamental issues to be addressed by the project such that they are considered in a professional and holistic manner while maintaining the patient's privacy and confidentiality. Ethical constraints are heavily regarded to ensure that the patient is always informed of whatever information is disclosed, if the patient is incapable of doing so an external and validated individual is to do so explicitly while informing the patient as well. In conclusion, this chapter aims to point out how these issues and constraints are to be handled, regarded and addressed responsibly.

6.0 SYSTEM ARCHITECTURE AND DESIGN

6.1 Chapter Overview

This chapter intends on covering the architecture regarded by the system (with each of its components explained in detail) while pointing out the design of it as well such that each component and its features are simplified such that it is easier to understand. The chapter includes a component diagram, class diagram, sequence diagram, instances of the UI with its UX elements defined as well. The illustrations are crucial to the contribution of information in this chapter

6.2 Design Goals

6.3 System Architecture Design

6.4 System Design

6.4.1 Choice of Design Paradigm

There are two primary system design methodologies.

1. SSADM - Structured System Analysis and Design Methodology

2. OOAD - Object Oriented Analysis and Design

The fall detection system is planned to be built with modular, reusable and scalable components. Therefore, the Object-Oriented Analysis and Design (OOAD) is the preferred design methodology for this system. OOAD can provide a structured approach to enhance maintainability and flexibility, which will be a proper way to build the fall detection system.

The system contains multiple components, they can be integrated into classes to follow the best object-oriented principles. Additionally, since the system integrates multiple sensor inputs (accelerometer, gyroscope, heart rate and SpO2 levels, and camera feed) to analyse and get real-time fall detection results, the system's complexity points out the need for efficient data processing, pipeline handling and efficient handling of other functionalities. The OOAD ensures that the system is structured properly allowing for better organization and reusability of code wherever needed.

Since Structures Systems Analysis and Design Methodology (SSADM, is a design paradigm that follows rigid sequential processes, here in this project OOAD is seen as a better option. As the user's needs vary and the technologies and tools get advanced, the fall detection system will have the need to expand itself. For that the SSADM's lack of flexibility in managing evolving system requirements can be a challenge. Given the dynamic nature of real-time sensor data and machine learning workflows, OOAD is the most suitable design methodology for this project, ensuring methodology for this suitable design methodology for this project.

6.4.2 Component Diagram

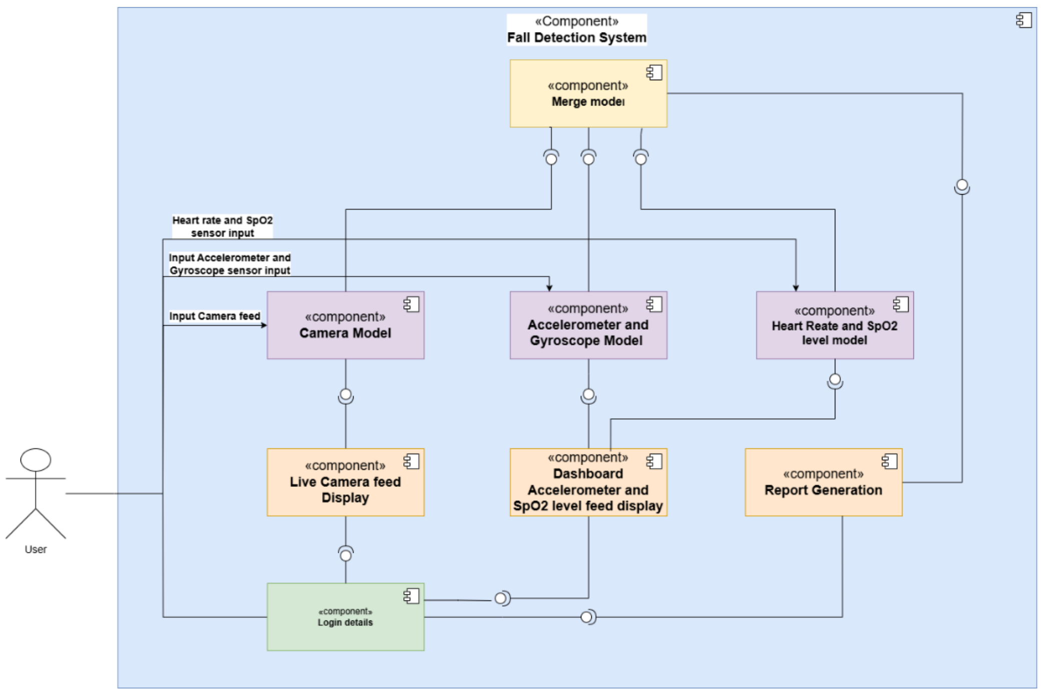


Figure 8: Component Diagram

6.4.3 Class Diagram

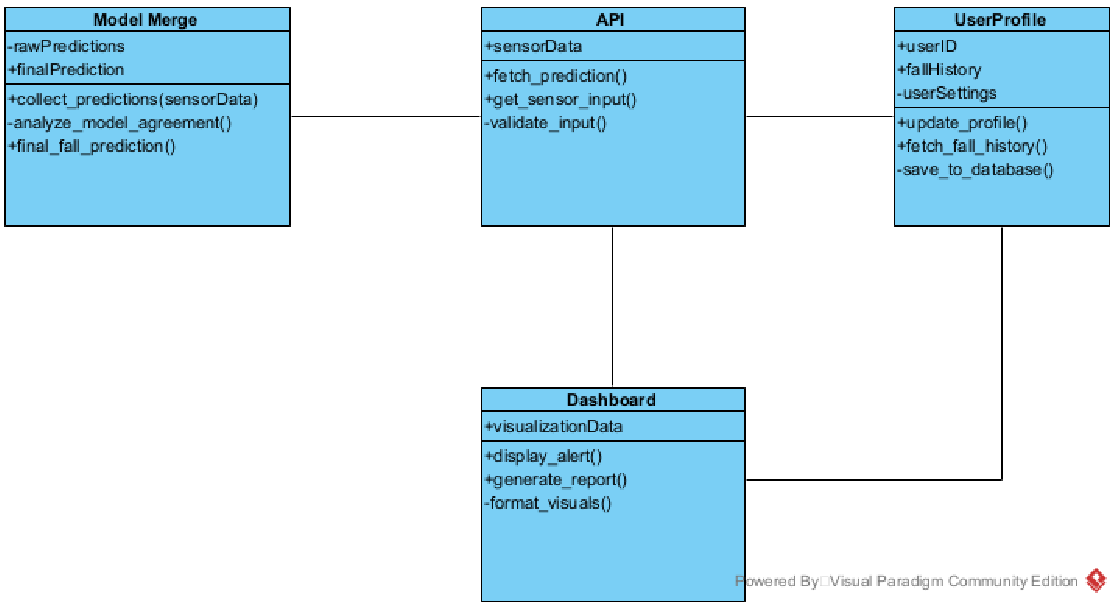


Figure 9: Class Diagram

6.4.4 Sequence Diagram

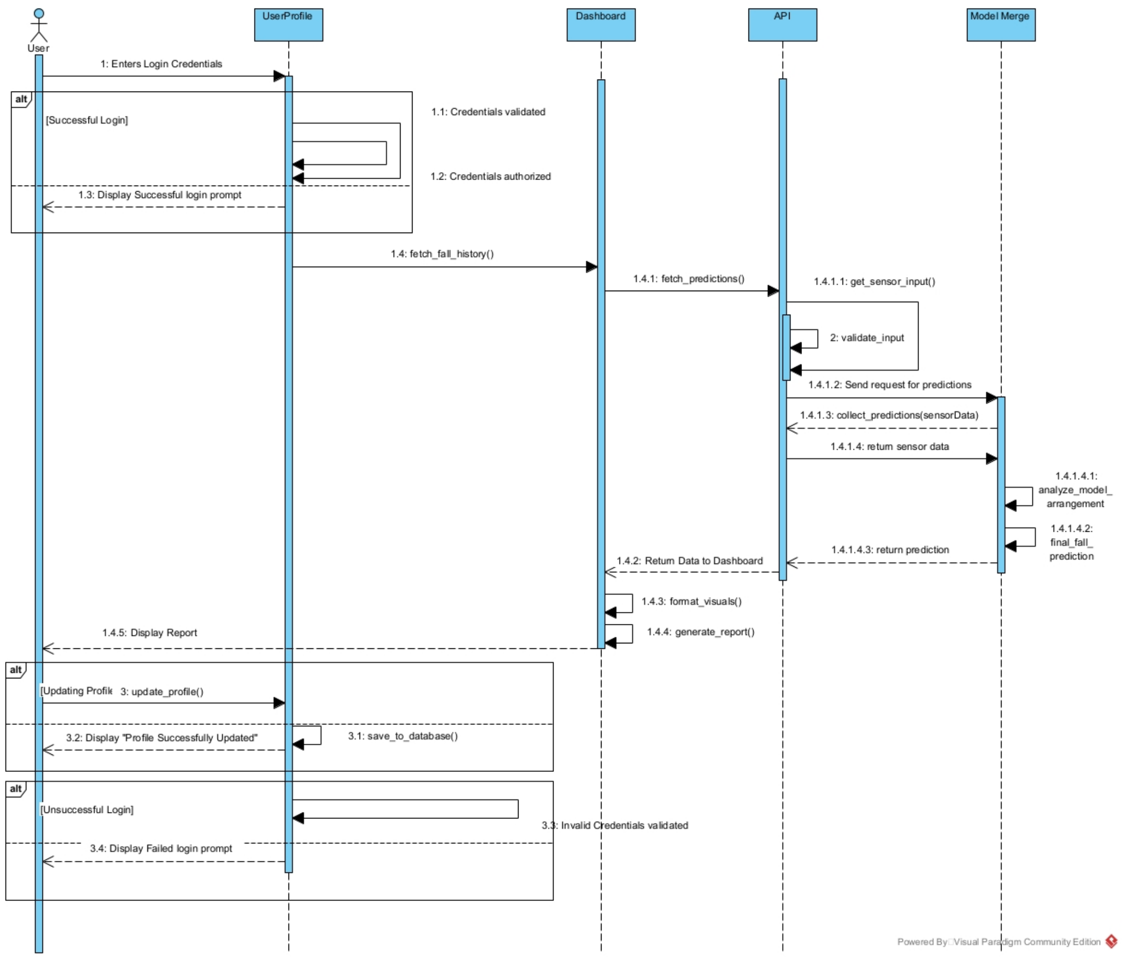
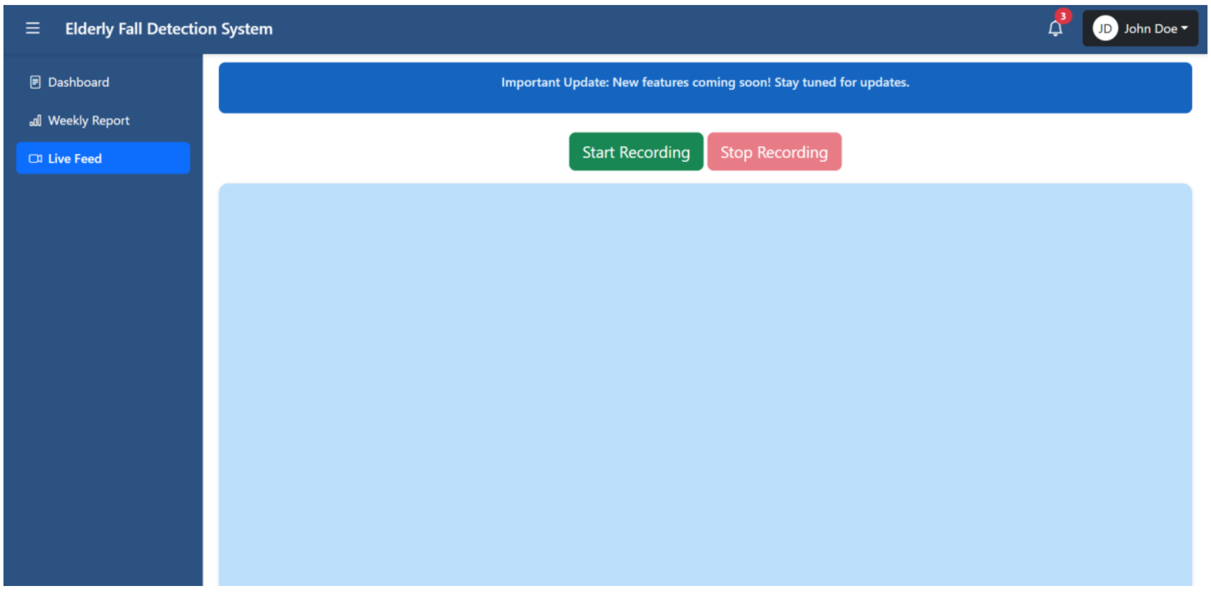

User
UserProfile
Dashboard
API
Model Merge
1: Enters Login Credentials
alt
[Successful Login]
1.1: Credentials validated
1.2: Credentials authorized
1.3: Display Successful login prompt
1.4: fetch_fall_history()
1.4.1: fetch_predictions()
1.4.1.1: get_sensor_input()
2: validate_input
1.4.1.2: Send request for predictions
1.4.1.3: collect_predictions(sensorData)
1.4.1.4: return sensor data
1.4.1.4.1: analyze_model_arrangement
1.4.1.4.2: final_fall_prediction
1.4.1.4.3: return prediction
1.4.2: Return Data to Dashboard
1.4.3: format_visuals()
1.4.4: generate_report()
1.4.5: Display Report
alt
[Updating Profile
3: update_profile()
3.1: save_to_database()
3.2: Display "Profile Successfully Updated"
alt
[Unsuccessful Login]
3.3: Invalid Credentials validated
3.4: Display Failed login prompt
Powered By: Visual Paradigm Community Edition

Figure 10: Fall Detection Sequence Diagram

6.4.5 UI Design



Important Update: New features coming soon! Stay tuned for updates.

Figure 11: Camera Viewing Page Layout

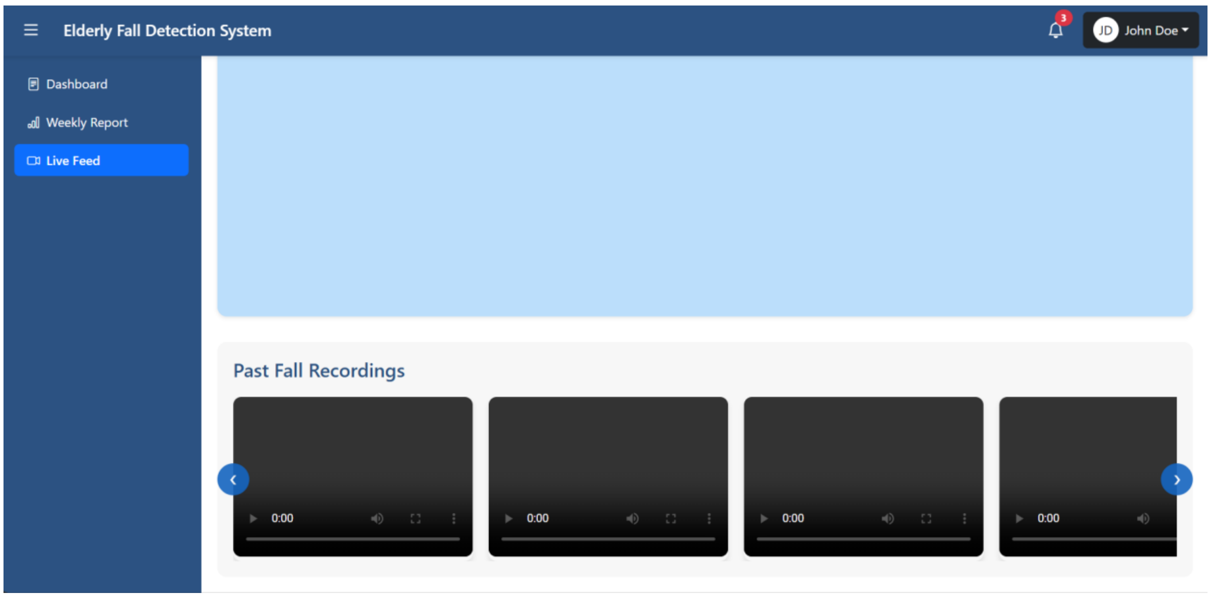
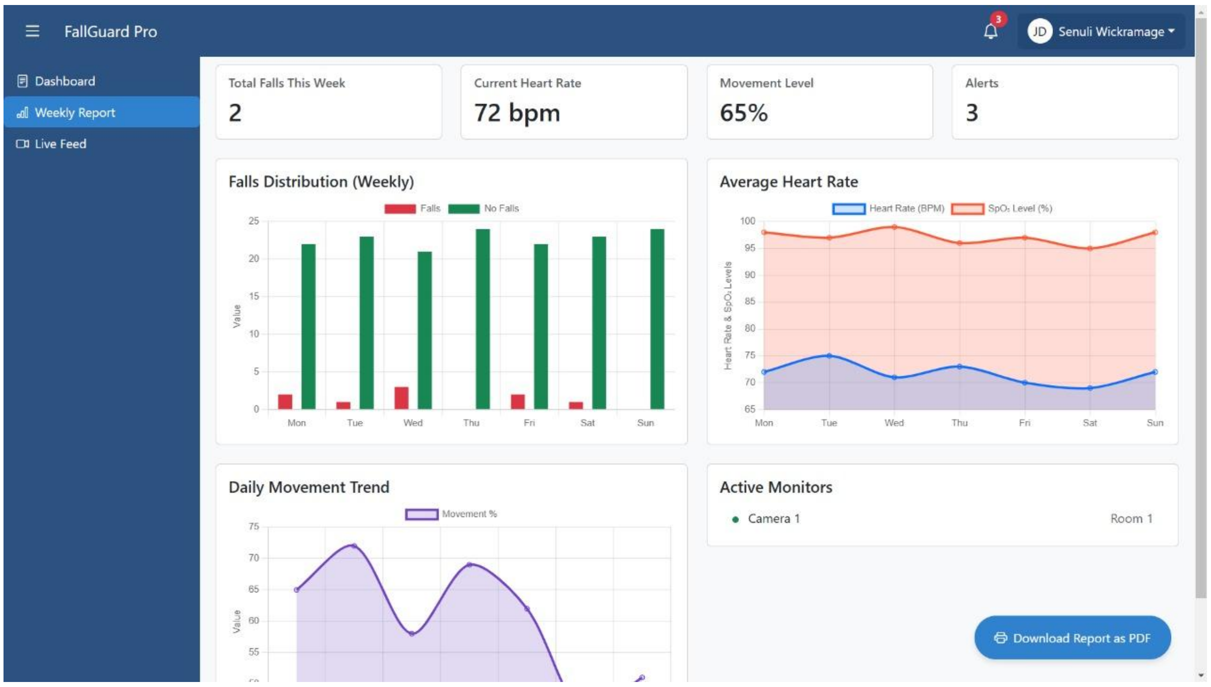


Figure 12: Past Recordings Section



Total Falls This Week 2

Current Heart Rate 72 bpm

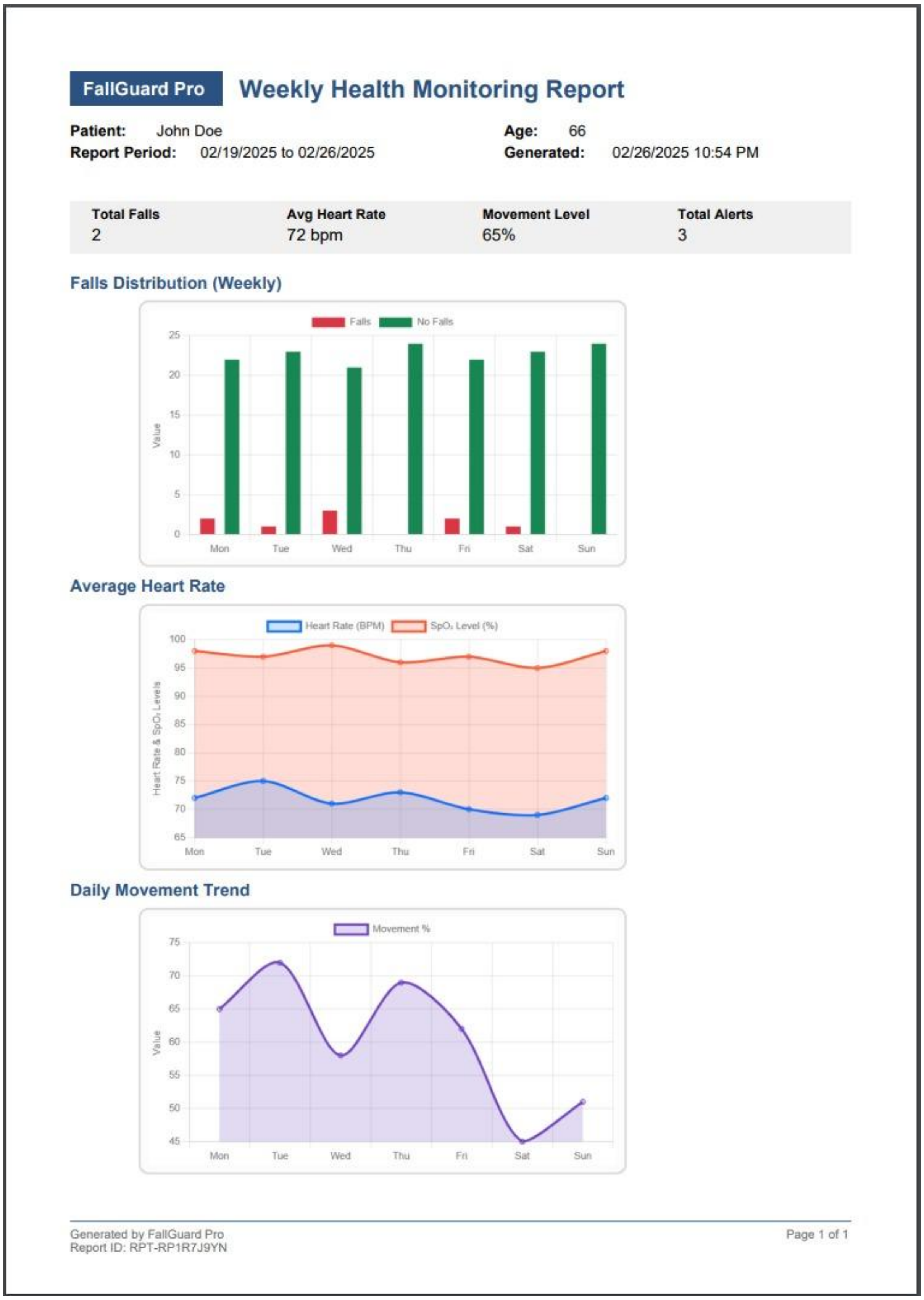
Movement Level 65%

Alerts 3

Active Monitors

● Camera 1 Room 1

Figure 13: Weekly Report Feed



FallGuard Pro **Weekly Health Monitoring Report**

Patient: John Doe **Age:** 66
Report Period: 02/19/2025 to 02/26/2025 **Generated:** 02/26/2025 10:54 PM

Total Falls	Avg Heart Rate	Movement Level	Total Alerts
2	72 bpm	65%	3

Falls Distribution (Weekly)

🟥 Falls
🟩 No Falls
Value
25
20
15
10
5
0
Mon
Tue
Wed
Thu
Fri
Sat
Sun

Average Heart Rate

Heart Rate (BPM)
SpO₂ Level (%)
Heart Rate & SpO₂ Levels
100
95
90
85
80
75
70
65
Mon
Tue
Wed
Thu
Fri
Sat
Sun

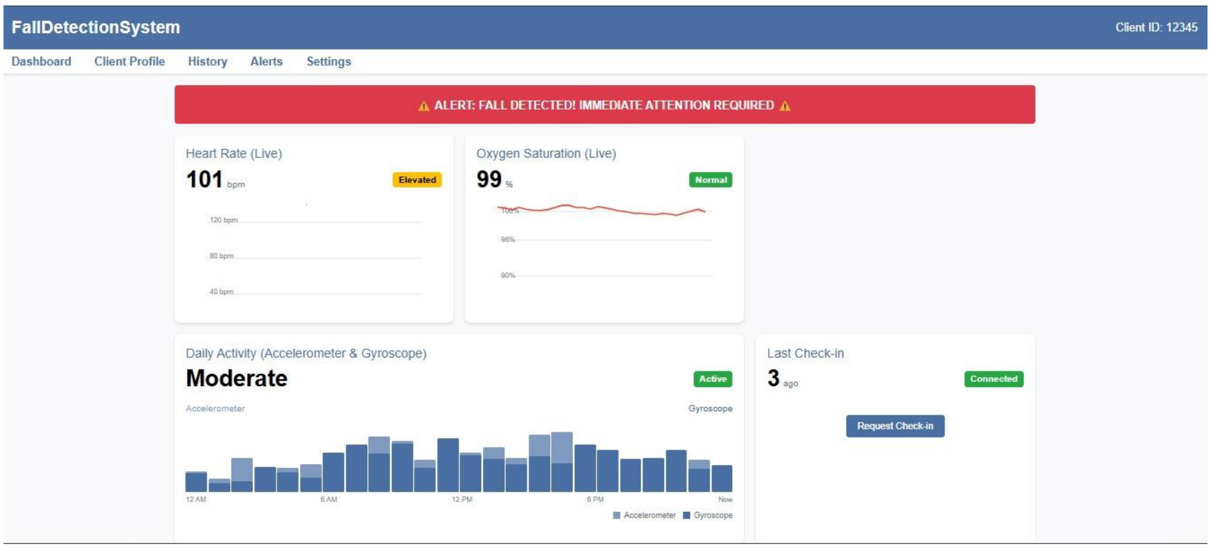
Daily Movement Trend

Movement %
Value
75
70
65
60
55
50
45
Mon
Tue
Wed
Thu
Fri
Sat
Sun

Generated by FallGuard Pro
Report ID: RPT-RP1R7J9YN

Page 1 of 1

Figure 14: Sample Weekly Report



⚠ ALERT: FALL DETECTED! IMMEDIATE ATTENTION REQUIRED ⚠

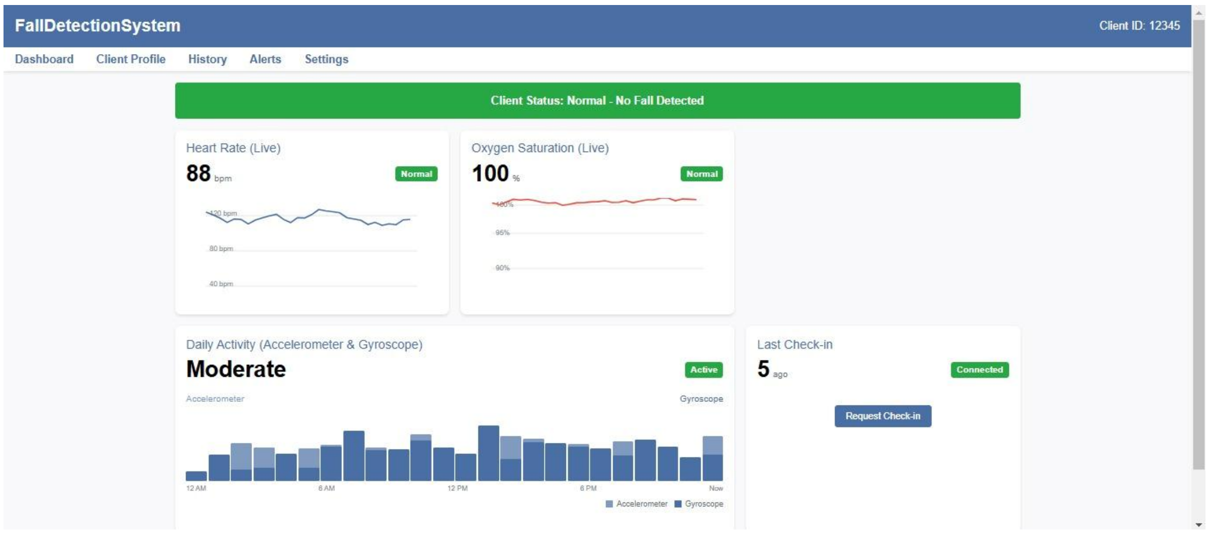
Heart Rate (Live) 101 bpm

Oxygen Saturation (Live) 99 %

Daily Activity (Accelerometer & Gyroscope) Moderate

Last Check-in 3 ago

Figure 15: Dashboard Page (With alert)



Client Status: Normal - No Fall Detected

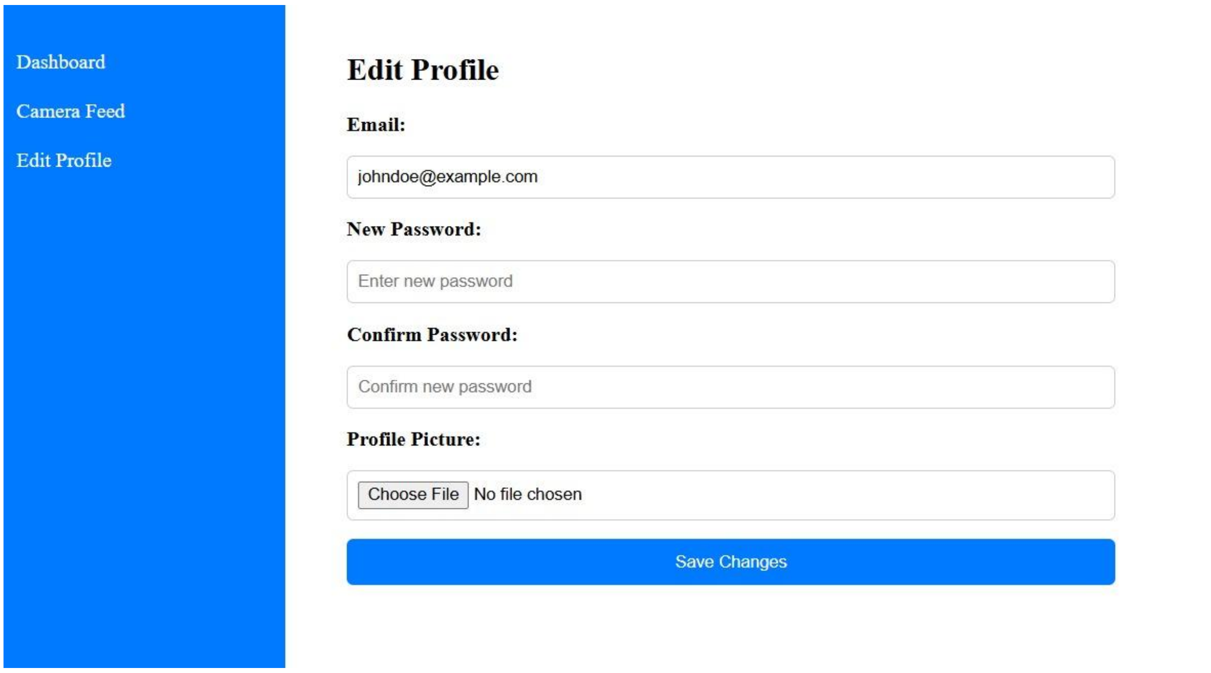
Heart Rate (Live) 88 bpm

Oxygen Saturation (Live) 100 %

Daily Activity (Accelerometer & Gyroscope) Moderate

Last Check-in 5 ago

Figure 16: Dashboard Page (Without alert)



Edit Profile

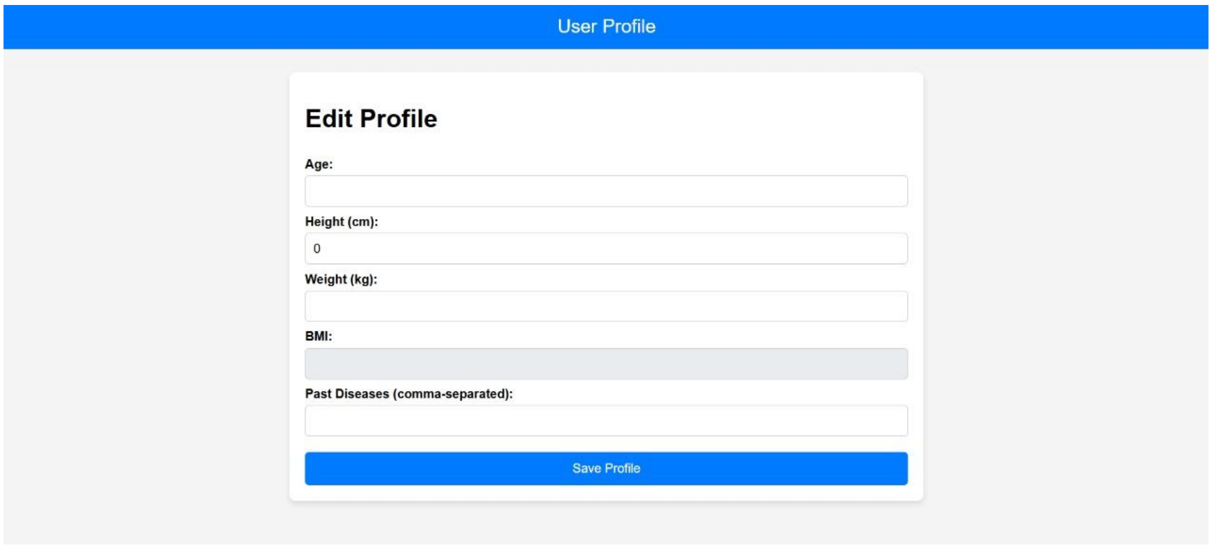
Email:
johndoe@example.com

New Password:
Enter new password

Confirm Password:
Confirm new password

Profile Picture:
Choose File No file chosen

Figure 17: User Profile Form



Edit Profile

Age:

Height (cm):
0

Weight (kg):

BMI:

Past Diseases (comma-separated):

Figure 18: Form for past disease

6.4.6 User Experience

Regarding the user experience, the application intends on catering to both elderly and new users alike such that it is easy for them to navigate through the website and access the required elements they need first. The dashboard is where the user is first placed in as it is where they may see an immediate overview of the real time data analysis, they would be able to see the patients/user's movements, heart rate and SP02 levels. More importantly, they can see an alert on the dashboard page that is initiated if the user is in risk of a fall.

Thereafter there is a navigation board that helps assists the user in navigating themselves to the right page to either get view the users through the camera feature of the application or view the users' weekly reports.

To ensure that the user retain an aesthetic experience that enables them to view and access each element with ease, the site has integrated bootstrap elements into it to minimize the load times of the webpage and maximize the ability to reach one page or another.

6.4.7 Process Flow Chart

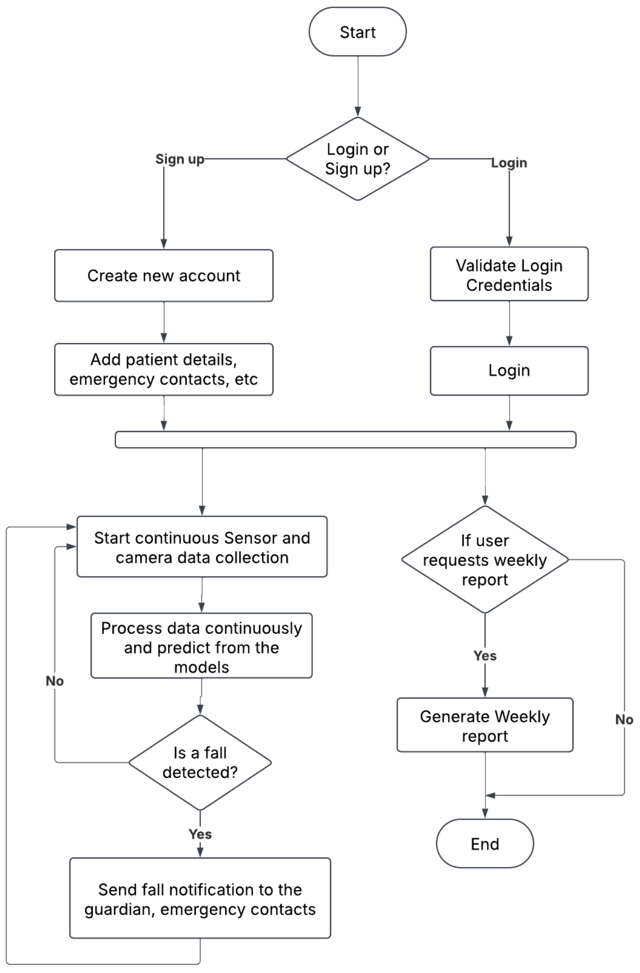

Start
Login or Sign up?
Sign up
Login
Create new account
Validate Login Credentials
Add patient details, emergency contacts, etc
Login
Start continuous Sensor and camera data collection
Process data continuously and predict from the models
No
Is a fall detected?
Yes
Send fall notification to the guardian, emergency contacts
If user requests weekly report
Yes
Generate Weekly report
No
End

Figure 19: Process Flowchart

6.5 Chapter Summary

Regarding the contents of this chapter, the design and architecture of the fall detection project was covered such that each components important features were covered such that they point out how the process of detecting a fall is regarded on a fundamental level. Besides that, the chapter covered the database/functional level of the project which was inclusive of how the user's profiles were handled in the application. To conclude, this chapter provided a concise overview of the structure of the application.

7.0 IMPLEMENTATION

7.1 Chapter Overview

This chapter intends on covering the implementations made to formalize the project into a functioning application. The chapter goes through the technology stacks used, frameworks, languages, libraries and other approaches taken to make a functional backend. Refer to the following sub sections to understand how this was done.

7.2 Technology Selections

7.2.1 Technology Stack

In terms of the technologies referred to develop the application, the workflow is as follows. The frontend was developed using VS-Code as the code editor and the languages referred to include HTML, CSS and JS. Bootstrap was used for consistent designs across each page. The backed was completely developed on Pycharm given its ease of access. The languages referred to include python and SQL. The frameworks include Django to host the entire application on an interconnected interface. Some of the main libraries referred to for the joint projections and model evaluations include, SciKitlearn and OpenCV.

Figure 20: Technology Stack

7.2.2 Data Selection

In terms of the data sets referred to by each model there was an evaluation process that deemed some datasets more useful than the other based on their information gain, feature relevance and context awareness. Some of these datasets were even catered to specifically fall-related collections, refer to the following:

Domain	Dataset	Desciption
Fall Detection based on motion sensor data (Accelerometer, Gyroscope)	Sisfall Dataset	The dataset includes triaxial accelerometer and gyroscope data collected at 200Hz from 38 participants performing 15 types of falls and 19 daily activities (ADLs). It features data from both young adults (19–30 years) and elderly participants (60–75 years), ensuring diverse motion patterns for model training. The dataset serves as a benchmark for fall detection research. Dataset Description – https://pmc.ncbi.nlm.nih.gov/articles/PMC5298771/ Link to the Dataset – https://www.kaggle.com/datasets/nvnikhil0001/sis-fall-original-dataset
Fall Detection Dataset based on motion captures and joint projections (joint based movements)	Fall Detection Dataset	The dataset consists of 374 images, each of which are either categorized as a fall or non-fall. A total of 111 images were presented for the validation process to evaluate the model's performance, however it was deemed unnecessary given the fact that a live feed was to be fed into the model. Each fall's coordinates were saved in the validations section, however, were not used given the different measuring criteria. Each fall is different from the other in each image and is provided with different environmental contexts. The provided dataset had however considered walking and sitting as an additional component as well, each fall state may be viewed in the following dataset: Dataset Description & Dataset – https://www.kaggle.com/datasets/uttejkumarkandagatla/fall-detection-dataset
Elderly Fall Prediction and detection A fall detection and prediction dataset for older elders.	cStick	The cStick.csv file contains sensor data used for fall detection and prediction. It includes features like distance, pressure (0: small, 1: medium, 2: high), HRV, sugar levels, SpO2, and accelerometer readings. The target variable, Decision, indicates fall status:

		• 0 – No fall detected • 1 – Fall prediction (slip/trip) • 2 – Definite fall This dataset helps improve fall prevention by identifying risks early. Link to the Dataset- https://www.kaggle.com/datasets/laavanya/elderly-fall-prediction-and-detection

Figure 21: Data Selection Table

7.2.3 Selection of Development Framework

Frontend

The frontend of the fall detection system was built using HTML, CSS and JavaScript. These technologies have provided a simple yet effective approach to creating an interacting user interface for the system. For the dynamic content update, real-time visualization of sensor data and to ensure a smooth user experience JavaScript has been utilized.

Backend

The backend was built using Python with one of its high-level web frameworks, Django. This has facilitated in efficient data processing, API development and seamless integration with machine learning models. The Django's built-in functionalities have ensured that the fall detection system has a robust backend with modular architecture, scalable, maintainable abilities. The final backend robustly enables real-time sensor data handling, communication between the frontend and backend.

7.2.4 Programming Language

In terms of the programming languages used in the fall detection system, Python was used as the main programming language in terms of model building and the backend of the project. To build all the functionalities to work efficiently a few libraries and frameworks in Python were used. For data analysis, data pre-processing and feature engineering popular libraries like Pandas, NumPy and Scikit-learn were used. The machine learning model in the motion sensor data component was built using libraries like TensorFlow and Keras. Moreover, for video processing, frame analysis and to handle real-time posture detection, libraries like OpenCV and Media-Pipe were used.

To serve more to the backend of the project, Python's very popular web framework Django was used. It has helped in managing areas like data processing, integration of machine learning models and handling API requests. Django's robust architecture has helped in ensuring the system to have a smooth connection between the frontend, backend and all functionalities to work efficiently.

The frontend of the application was built using HTML, CSS and JavaScript. These technologies have provided a well working, responsive, and a user-friendly interface for the system, ensuring that the users have a great time using the system. The frontend interface also provides real-time visualization of sensor data and fall detection results. In Addition to that JavaScript was used to enhance the interactivity and to ensure the smooth updates without loading the full page.

The fall detection system has gained efficient working ability due to the combination of these technologies in the right way. In all the functionalities like handling incoming data, applying to ML models, and in delivering the real time fall detection results and insights through the web application, it can be considered that the backbone made of these technologies have played an immense role.

7.2.5 Libraries

Libraries	**Version**
SciKitLearn	1.6.1
Keras	3.9.0
Numpy	2.2.3
Mediapipe	0.10.21
OpenCV	4.11.0
Matplotlib	3.10.1
Joblib	1.4.2
Pickle	0.0.12

Table 22: Libraries used

7.2.6 Integrated Development Environment (IDE)

For the development of the fall detection system, **PyCharm** and **Visual Studio Code (VS Code)** were the primary Integrated Development Environments (IDEs) used. These IDEs provided efficient tools for coding, debugging, and seamless integration between the backend and frontend components.

PyCharm for Backend Development

PyCharm was used to develop the backend of the application, particularly for **Django-based development**. It provided features such as **intelligent code completion, debugging tools, virtual environment management, and built-in support for Django**. These capabilities streamlined the implementation of backend functionalities, including **handling API requests, integrating machine learning models, and managing databases**. Additionally, PyCharm's structured interface facilitated efficient organization of the project files and modules.

VS Code for Frontend Development

VS Code was utilized for designing and implementing the frontend using **HTML, CSS, and JavaScript**. Its lightweight nature and **customizable extensions** allowed developers to work efficiently on the user interface. The **Live Server extension** enabled real-time previewing of frontend changes, ensuring a dynamic and responsive design. Furthermore, VS Code’s **Git integration** provided version control, making it easier to track changes and collaborate on the project.

By combining **PyCharm for backend development** and **VS Code for frontend implementation**, the fall detection system was developed efficiently, ensuring seamless interaction between the **Django-powered backend** and the **web-based user interface**. These IDEs played a crucial role in the successful implementation and optimization of the system.

7.2.7 Summary of Technology Selection

Component	Technology/Tool	Version
Programming Language	Python	3.14
UI Framework	Django	5.1.7
	Bootstrap	5.3
IDE	Visual Studio Code	1.98
	Pycharm	2024.3.2
	Google Collab	-

Table 23: Summary of technology selection

7.3 Implementation of Core Functionalities

7.3.1. Accelerometer, Gyroscope sensor data-based fall detection model

Fall Detection System Logic

```
Input: serial_port, baud_rate
Output: fall_detection_status

Initialize latest_sensor_data
latest_sensor_data ← {accelerometer: {x: 0, y: 0, z: 0}, gyroscope: {x: 0, y: 0, z: 0}, fall_detected: false, timestamp: current_time}

FUNCTION FallDetectionSystem(serial_port, baud_rate)
    Initialize system variables
    serial_connection ← NULL
    is_running ← FALSE
    model ← load("fall_detection_model.pkl")
    scaler ← load("acc_gyro_scaler.pkl")
    segment_length ← 1000
    current_frame ← []
```

```
FUNCTION connect_to_arduino()
    TRY
        serial_connection ← open_serial(serial_port, baud_rate)
        RETURN TRUE
    CATCH error
        PRINT "Failed to connect to Arduino"
        RETURN FALSE
    END TRY
END FUNCTION

FUNCTION read_sensor_data()
    IF serial_connection IS NULL THEN
        RETURN NULL
    END IF

    TRY
        line ← serial_connection.readline()
        values ← parse_to_float_list(line)

        IF length(values) = 6 THEN
            sensor_data ← {
                accelerometer: {x: values[0], y: values[1], z: values[2]},
                gyroscope: {x: values[3], y: values[4], z: values[5]},
                raw_values: values,
                timestamp: current_time
            }
            RETURN sensor_data
        ELSE
            PRINT "Invalid data format"
            RETURN NULL
        END IF
    CATCH error
        PRINT "Error reading sensor data"
        RETURN NULL
    END TRY
END FUNCTION

FUNCTION process_time_frame(frame)
    IF length(frame) < segment_length THEN
        padding ← segment_length - length(frame)
        frame ← frame + [last_element(frame)] * padding
    ELSE IF length(frame) > segment_length THEN
        frame ← frame[0:segment_length]
    END IF

    frame_array ← create_array_from_raw_values(frame)
    X ← reshape(frame_array, [1, segment_length, 6])
    X_reshaped ← reshape(X, [-1, 6])
    X_scaled ← scaler.transform(X_reshaped)
    X_scaled ← reshape(X_scaled, [1, segment_length, 6])

    prediction ← model.predict(X_scaled)[0]

    latest_sensor_data.fall_detected ← boolean(prediction)

    RETURN boolean(prediction)
END FUNCTION

FUNCTION start_monitoring()
    IF NOT connect_to_arduino() THEN
        RETURN FALSE
    END IF

    is_running ← TRUE
    current_frame ← []

    PRINT "Starting fall detection monitoring..."

    WHILE is_running = TRUE DO
        sensor_data ← read_sensor_data()

        IF sensor_data IS NOT NULL THEN
            latest_sensor_data.accelerometer ← sensor_data.accelerometer
            latest_sensor_data.gyroscope ← sensor_data.gyroscope
            latest_sensor_data.timestamp ← sensor_data.timestamp

            append(current_frame, sensor_data)

            IF length(current_frame) >= segment_length THEN
                fall_detected ← process_time_frame(current_frame)

                IF fall_detected = TRUE THEN
                    PRINT "FALL DETECTED!"
                END IF

                current_frame ← current_frame[segment_length/2:]
            END IF
        END IF
    END WHILE

    IF serial_connection IS NOT NULL THEN
        close(serial_connection)
    END IF
END FUNCTION

FUNCTION stop_monitoring()
```

```
        is_running ← FALSE
    END FUNCTION
END FUNCTION

FUNCTION get_sensor_data(request)
    RETURN format_as_json(latest_sensor_data)
END FUNCTION

FUNCTION handle_websocket(websocket)
    WHILE TRUE DO
        websocket.send(convert_to_json(latest_sensor_data))
        sleep(0.05)
    END WHILE
END FUNCTION

fall_detector ← FallDetectionSystem(serial_port, baud_rate)
start_thread(fall_detector.start_monitoring)

FUNCTION main()
    fall_detector ← FallDetectionSystem(serial_port, baud_rate)
    TRY
        fall_detector.start_monitoring()
    CATCH keyboard_interrupt
        fall_detector.stop_monitoring()
        PRINT "Monitoring stopped"
    END TRY
END FUNCTION
```

7.3.2 Joint Based Fall Detection model

```
BEGIN
    FUNCTION get_camera_page(request)
        recordings_dir ← concatenate(settings.MEDIA_ROOT, "/", "recordings")

        video_files ← EMPTY_LIST
        IF path_exists(recordings_dir) THEN
            FOR each file in list_directory(recordings_dir) DO
                IF file ends_with(".mp4") OR file ends_with(".avi") OR file ends_with(".mov")
THEN
                    append(video_files, file)
                END IF
            END FOR
        END IF

        context ← { "videos": video_files }

        RETURN render_template(request, "camera_feed.html", context)
    END FUNCTION
```

```
FUNCTION gen_frames()
    base_dir ← get_parent_directory(get_parent_directory(get_absolute_path(__file__)))
        model_dir ← concatenate_paths(settings.BASE_DIR, "fallguard_pro", "static",
"KNN_model")
    scaler_path ← concatenate_paths(model_dir, "scaler.pkl")
    model_path ← concatenate_paths(model_dir, "knn_model.pkl")

    TRY
        scaler ← load_model(scaler_path)
        model_file ← open_file(model_path, "rb")
        knn_model ← load_pickle(model_file)
    CATCH error
        PRINT "Error loading model: ", error
        YIELD NULL
        RETURN
    END TRY

    mp_pose ← initialize_pose_module()
    mp_drawing ← initialize_drawing_utils()
    non_face_indices ← [11 to 32]

    cap ← open_camera(0)
    IF camera_not_opened(cap) THEN
        PRINT "Error: Could not open camera."
        RETURN
    END IF

    recordings_dir ← concatenate_paths(settings.MEDIA_ROOT, "recordings")
    create_directory_if_not_exists(recordings_dir)

    log_filename ← "fall_detection_log.csv"
    log_path ← concatenate_paths(settings.MEDIA_ROOT, log_filename)
    create_directory_if_not_exists(get_parent_directory(log_path))

    log_file ← open_file(log_path, "w")
    writer ← create_csv_writer(log_file)
    write_csv_row(writer, ["Timestamp", "Status"])

    fall_start_time ← NULL
    non_fall_start_time ← get_current_time()
    recording ← FALSE
    video_filename ← NULL
    fall_video_writer ← NULL
    fourcc ← define_video_codec("mp4v")

    pose ← initialize_pose(min_detection_confidence=0.3, min_tracking_confidence=0.3)
    WHILE TRUE
        success, frame ← read_frame(cap)
        IF success IS FALSE THEN
            BREAK
        END IF

        image ← resize_frame(frame, width=640, height=480)
        image_rgb ← convert_to_rgb(image)
        results ← process_pose(pose, image_rgb)

        label ← "No Pose Detected"
        IF pose_landmarks_detected(results) THEN
            height, width, _ ← get_frame_dimensions(image)
            landmarks ← get_pose_landmarks(results)
            filtered_landmarks ← filter_landmarks(landmarks, non_face_indices)

            x_coords ← extract_x_coordinates(filtered_landmarks, width)
            y_coords ← extract_y_coordinates(filtered_landmarks, height)
            visibilities ← extract_visibilities(filtered_landmarks)

            x_min, x_max ← min(x_coords), max(x_coords)
            y_min, y_max ← min(y_coords), max(y_coords)

            features_input ← [
                mean(x_coords),
                mean(y_coords),
                mean(visibilities),
                x_max - x_min,
                y_max - y_min
            ]

            features_input_scaled ← transform_with_scaler(scaler, [features_input])
            prediction ← predict_with_model(knn_model, features_input_scaled)[0]
            label ← choose_label(prediction, "Falling", "Not Falling")

            current_time ← get_current_datetime_string()
            write_csv_row(writer, [current_time, label])

            IF prediction = 1 THEN
                IF fall_start_time IS NULL THEN
                    fall_start_time ← get_current_time()
                    timestamp_str ← get_formatted_timestamp()
                    video_filename ← concatenate_paths(recordings_dir, concatenate_strings("fall_", timestamp_str, ".mp4"))
                    fall_video_writer ← create_video_writer(video_filename, fourcc, 20.0, (640, 480))
                    recording ← TRUE
                END IF

                IF recording IS TRUE THEN
                    write_frame_to_video(fall_video_writer, image)
                    END IF
                ELSE
                    IF fall_start_time IS NOT NULL THEN
                        fall_duration ← calculate_time_difference(fall_start_time)
                        IF fall_duration < 5 AND recording IS TRUE THEN
                            release_video_writer(fall_video_writer)
                            delete_file(video_filename)
                        ELSE IF fall_duration ≥ 5 AND recording IS TRUE THEN
                            release_video_writer(fall_video_writer)
                        END IF
                        fall_start_time ← NULL
                        recording ← FALSE
                    END IF
                END IF

                draw_landmarks_on_image(image, results, mp_drawing, mp_pose)
                draw_bounding_box(image, x_min, y_min, x_max, y_max, color=(0, 255, 0))
                  draw_text_on_image(image, label, position=(50, 50), font_size=1, color=(0, 0,
255))

        ret, buffer ← encode_frame(image, format=".jpg")
        frame_bytes ← get_bytes_from_buffer(buffer)
             YIELD  concatenate_bytes("--frame\r\n",  "Content-Type:  image/jpeg\r\n\r\n",
frame_bytes, "\r\n")
    END WHILE

    release_camera(cap)
    IF fall_video_writer IS NOT NULL THEN
        release_video_writer(fall_video_writer)
    END IF

    PRINT concatenate_strings("Fall detection log saved to: ", log_path)
  END FUNCTION

  FUNCTION camera_feed(request)
    RETURN StreamingHttpResponse(
        gen_frames(),
        content_type = "multipart/x-mixed-replace; boundary=frame"
    )
  END FUNCTION

STOP
```

7.3.3 Heart Rate and SP02 Level based model

```
START
```

FUNCTION LoadAndPreprocessData → DataFrame (X, y)

LOAD CSV file into DataFrame STRIP spaces from column names IF required columns ('HRV', 'SpO2', 'Decision') are missing → RAISE ERROR REMOVE rows with missing values in ('HRV', 'SpO2', 'Decision') SET X ← ['HRV', 'SpO2'] SET y ← 'Decision' SPLIT X, y into (X_train, X_test, y_train, y_test) using train_test_split (80/20) RETURN X_train, X_test, y_train, y_test END FUNCTION

```
 --------------------------------------------------

FUNCTION StandardizeData(X_train, X_test) → X_train_scaled,
X_test_scaled, scaler
    INITIALIZE StandardScaler
    FIT StandardScaler on X_train
    TRANSFORM X_train and X_test using fitted StandardScaler
RETURN X_train_scaled, X_test_scaled, scaler
END FUNCTION

--------------------------------------------------

FUNCTION TrainKNNAndEvaluate(X_train_scaled, y_train,
X_test_scaled, y_test) → best_k, best_model
    SET k_values ← [3, 5, 7, 10]
    FOR each k in k_values → DO
        INITIALIZE KNeighborsClassifier with n_neighbors = k
        TRAIN model on (X_train_scaled, y_train)
        PREDICT y_pred on X_test_scaled
        CALCULATE accuracy and cross-validation score
        DISPLAY classification report and confusion matrix
    END FOR
    CHOOSE best_k based on highest accuracy
    RETRAIN model using best_k
RETURN best_k, best_model
END FUNCTION

--------------------------------------------------

FUNCTION SaveModelAndScaler(model, scaler, model_path,
scaler_path)
    SAVE model to 'knn_model.pkl'
    SAVE scaler to 'scaler.pkl'
END FUNCTION

--------------------------------------------------

FUNCTION LoadAndPredict(input_data, model_path, scaler_path) →
prediction
    LOAD model from 'knn_model.pkl'
    LOAD scaler from 'scaler.pkl'
    CONVERT input_data to DataFrame with column names ['HRV',
'SpO2']
    TRANSFORM input_data using loaded scaler
    PREDICT decision using loaded model
RETURN prediction
END FUNCTION

--------------------------------------------------

# MAIN PROGRAM EXECUTION
X_train, X_test, y_train, y_test ← LoadAndPreprocessData()
X_train_scaled, X_test_scaled, scaler ← StandardizeData(X_train,
X_test)
best_k, best_model ← TrainKNNAndEvaluate(X_train_scaled, y_train,
X_test_scaled, y_test)
SaveModelAndScaler(best_model, scaler, 'knn_model.pkl',
'scaler.pkl')

# Example Usage for Prediction
sample_input ← [[50, 95]]
predicted_decision ← LoadAndPredict(sample_input, 'knn_model.pkl',
'scaler.pkl')
DISPLAY predicted_decision


END
```

7.4 Chapter Summary

This chapter had covered the technologies referred to develop the application in terms of its coding backend, database approaches and other means of training the models and developing the frontend. The core functionalities of each model were listed out as well, covering how each of them functions in terms of assessing a fall. The diagram in the technology stack presents the front end and backend components included in the development of the application, each of which show how the more core and important components of the application were defined and made.

8.0 TESTING

8.1 Chapter Overview

In this chapter, the performance metrics of each machine learning model in the project is evaluated, assessed and tested such that they each meet a certain standard before being deployed in an active environment. The functional tests, benchmarking, module and integration testing as well as non-functional testing components where load balancing, accuracy assessment and performance evaluations are carried out are explained here.

8.2 Objectives and Goals of Testing

The primary objective of the tests to be carried out is to assess if the application is suitable for an industry standard environment such that it may satisfy the needs of each patient with a recursive performance metric. The main goals of the testing phase are to:

- To identify potential issues within the system such as false positives for certain detections and assessments of how the environment could obstruct the applications performance
- To evaluate if the system performs as intended when deployed in a patient's environment
- To assess the model's performance by measuring the total number of false positives and true positives detected (vice versa for negatives) such that the model's accuracy is evaluated
- To see if there are any potential vulnerabilities within the system that could disrupt and breach the patient's personal data

In conclusion, the process of testing the system for any vulnerabilities or performance issues can enhance the system while marking out any issues the system may have.

8.3 Testing Criteria

8.3.1 Functionality Testing Criteria

Sensor Data Processing and Integration:

Verify that the system correctly captures, processes, and integrates data from multiple sensors, including the **MPU6050 (accelerometer & gyroscope), SpO2 sensor, and camera feed**.

Real-Time Fall Detection:

Test whether the system can accurately detect falls in real-time based on the **individual model outputs** and the **final weighted voting approach**. Ensure the system correctly identifies falls, non-falls, and the "risk of falling" state.

Model Accuracy and Predictions:

Validate that each detection model contributes effectively to the final decision-making process:

- **Camera Model:** Assess its ability to identify falls based on body posture analysis using MediaPipe Pose and a KNN classifier.
- **Accelerometer & Gyroscope Model:** Test its effectiveness in detecting sudden changes in motion that indicate a fall event.
- **Heart Rate & SpO2 Model:** Ensure it accurately tracks physiological data and detects anomalies that may correlate with falls.

Weighted Model Fusion:

Ensure that the weighted voting mechanism correctly prioritizes the models, where the **camera model has the highest priority (weight = 4)**, followed by the **accelerometer & gyroscope model (weight = 2)**, and finally, the **heart rate & SpO2 model (weight = 1)**. Confirm that the final decision aligns with the expected behavior based on assigned weights.

System Alerts and Notifications:

Confirm that the system appropriately triggers alerts and notifications when a fall is detected, providing timely feedback for potential emergency response.

User Profile and Dashboard Functionality:

Test the functionality of the **UserProfile and Dashboard** components, ensuring users can view past fall events, sensor data history, and system-generated insights.

8.3.2 Non-Functionality Testing Criteria

Performance:

Assess the system's ability to process sensor data and make real-time predictions efficiently. Measure the response time for each model and for the overall system when detecting falls and generating alerts, ensuring minimal latency.

Reliability:

Test the system under varying conditions:

- Different user movements and lighting conditions (for the camera model)
- Different motion patterns and orientations (for the accelerometer & gyroscope model)
- Variability in physiological readings (for the heart rate & SpO2 model)

Ensure consistent performance and reliable fall detection across all conditions.

Security:

Ensure that **user data, including sensor readings and stored fall history, is securely processed and stored**. Verify that **data transmission between hardware and the system is encrypted** to prevent unauthorized access.

Scalability:

Analyse the system's capability to handle multiple users simultaneously, testing how it performs with increasing data loads and concurrent device connections.

Maintainability:

Review the **code structure, documentation, and modularity** to ensure that future updates, bug fixes, and enhancements can be implemented with minimal effort. Validate that the API interactions and model components are well-documented.

Energy Efficiency (for Wearable/Embedded Devices):

Evaluate power consumption to ensure that the system can operate efficiently on battery-powered devices, considering factors like **sensor polling frequency and computational load**.

8.4 Model Evaluation

To evaluate the models there are a variety of metrics to be considered. This mainly includes the classification report produced after the creation of a model once fitted with data and then run. The classification report points out the following parameters:

- Precision - measures the proportion of correctly predicted positive observations to the total predicted positives

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives (FP)}$$

Equation 1: Equation to measure Precision

- Recall (Sensitivity or True Positive Rate) - measures the proportion of correctly predicted positive observations to all actual positives

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives (TP) + False\ Negatives\ (FN)}$$

Equation 2: Equation to measure Recall

- F1-Score – Measures the harmonic mean of precision and recall by balancing the two metrics (in other words, measures the imbalance between the two classes)

$$F1\ Score = 2\ .\ \frac{Precision\ .\ Recall}{Precision\ +\ Recall}$$

Equation 3: Equation for F1-Score

- Support Vectors – Refers to the number of true instances for each class in the dataset
- Accuracy - measures the overall correctness of the model by dividing the number of correct predictions by the total predictions

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions}$$

Equation 4: Equation for Accuracy

- Macro Average - calculates the average of precision, recall, and F1-score across all classes, treating each class equally
- Weighted Average - calculates the average of precision, recall, and F1-score while taking class support into account

Thereafter regarding the confusion matrix, it simply points out the total number of true positives against the total number to true negatives and vice versa. This shows the following:

- True Positives – Total number of correct/true positive predictions made (e.g. Person A did fall)
- False Positives Total number of incorrect/false positive predictions made
- True Negatives – Total number of correct/true negative predictions made (e.g. Person A did not fall)
- False Negatives - Total number of incorrect/false negative predictions made

8.4.1 Classification Report

8.4.1.1 Joint Based Model

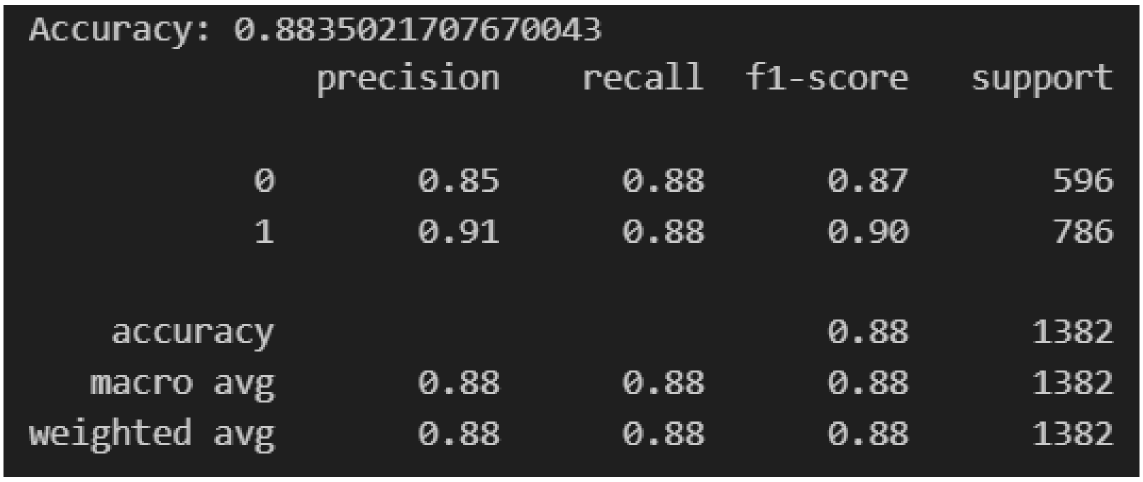
To detect whether a fall occurs through a joint based model a cross-validation score was carried out to finalize which model was the most ideal and optimal for the dataset created through the fall detection images that were used from the Kaggle based dataset. The models assessed were:

Model	Cross Validation Mean Accuracy	Model Accuracy	Verdict
Logistic Regression	0.7570	0.7641	• Model Performance too poor • Accuracy too low • Fails to detect falls well enough
Support Vector Classifier	0.8194	0.8119	• Model Performance Optimal but could be better • Accuracy not high enough
Random Forest	0.9935	0.9964	• Accuracy too high, hence leading to overfitting issues • Model better detects data found in the dataset that newly created data
K-Nearest-Neighbours	0.8844	0.8835	• Optimal Accuracy • Performance in task environment is credible as well

Table 24: Cross Validation Assessment for Joint Based Model

According to the outcomes of the table above it may be denoted that the most optimal model for the joint based model's predictions would be the KNN model in which had an accuracy above 85% and below 90%, hence making it ideal. There is a smaller risk of experiencing overfitting issues given the fact that the models overall accuracy is below 90%.

Before finalizing the model, a tuning was carried out to fine tune the model to address which hyper parameters were the most optimal for the provided dataset. It was then denoted that the ideal hyperparameters were 5 neighbours as it had retained an accuracy just shy of 90%. To conclude, the model's architecture was engineered and optimized for the provided created dataset, hence making it the most optimal to make predictions. The classification report was like so:


```
Accuracy: 0.8835021707670043
              precision    recall  f1-score   support

           0       0.85      0.88      0.87       596
           1       0.91      0.88      0.90       786

    accuracy                           0.88      1382
   macro avg       0.88      0.88      0.88      1382
weighted avg       0.88      0.88      0.88      1382
```

Figure 22: Classification Report Joint Based Model

8.4.1.2 Motion Sensor Data (Accelerometer and Gyroscope) Based Model

Three models were trained with the Sisfall dataset to compare and choose one model to carry on with. A hyper parameter tuning was done with grid searches for each 3 of these models. The given below shows the best accuracy of the three models with the best parameters.

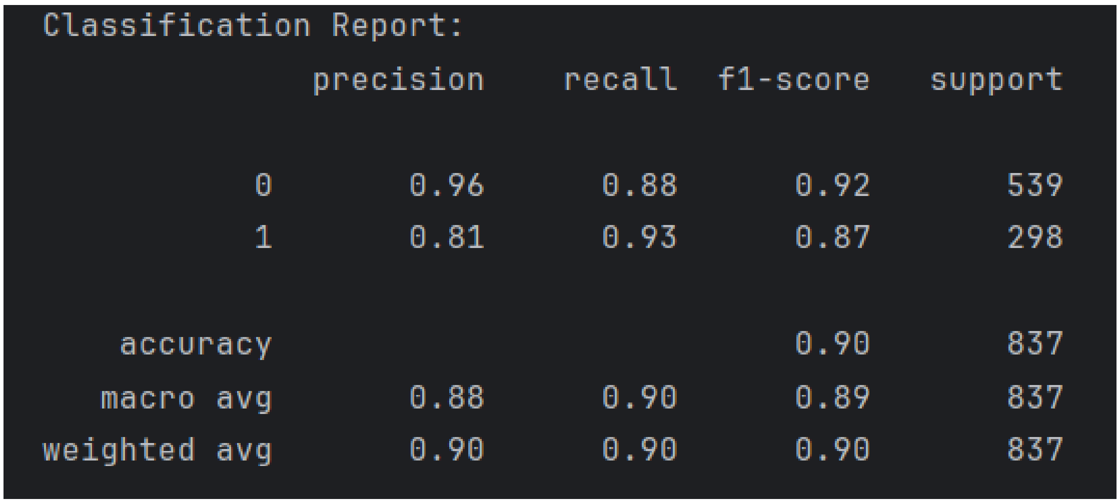
Model	Metrics	Verdict
CNN (Convolutional Neural Network)	**Accuracy**: 0.90 **Precision:** 0.96 0.81 **Recall:** 0.88, 0.93 **F1 Score:** 0.92, 0.87	• Model performance is good. • Captures patterns in the time-series data effectively. • Able to work with 3-dimensional data directly.
XGBoost	**Accuracy**: 0.95 **Precision**: 0.94, 0.98 **Recall**: 0.99, 0.88 **F1 Score**: 0.96, 0.93	• Need to convert 3D data into 2D. • Higher accuracy might be due to this transformation. • Possible loss of information, making it less ideal for sensor data.
TCN (Temporal Convolutional Network)	**Accuracy**: 0.90 **Precision**: 0.94, 0.83 **Recall**: 0.90, 0.90 **F1 Score**: 0.92, 0.86	• Also good for time series data, since TCN is an algorithm specific for time series data. • Similar performance to CNN, but CNN is a more standard approach.

Table 25: Model Assessment for Motion Sensor Data Model

Since the data fed into the models take a three-dimensional format which is structured as (sample, timesteps, features). Here, each sample consists of multiple time steps, and each time step contains six features: three from the accelerometer (x-axis, y-axis, z-axis) and three from the gyroscope.

CNN was chosen as the final model because it can process raw 3D sensor data without the need for converting the 3D data into 2D. On the other hand, XG Boost which is a machine learning model, cannot work directly from raw data. It requires converting this structured time-series data into a 2D format. CNN can directly learn special and temporal patterns from the original structure. CNN offers a more natural way to process raw sensor data in real-time while maintaining the meaning, which makes it the preferred choice for the motion sensor data fall detection model.

Given below is classification report of the final model – CNN



```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.88      0.92       539
           1       0.81      0.93      0.87       298

    accuracy                           0.90       837
   macro avg       0.88      0.90      0.89       837
weighted avg       0.90      0.90      0.90       837
```

Table 26: Classification Report - Motion Sensor Data Model

The below given two plots show how the model has improved its accuracy and decreased its validation during training and validation throughout the epochs. These figures show that the model has been identifying patterns well while not getting prone to overfitting.

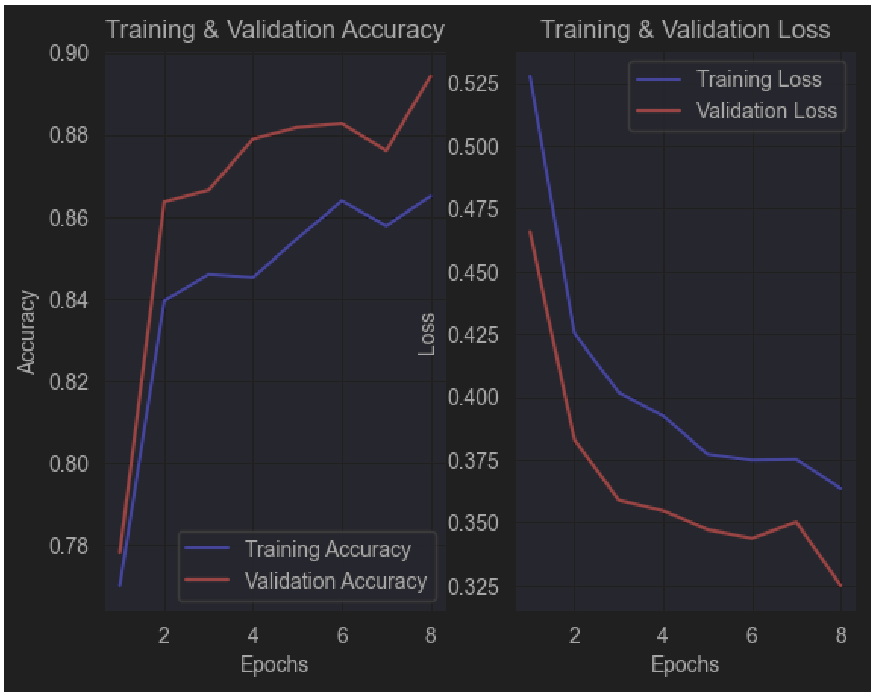


Table 27: Accuracy and Loss over the epochs - Motion Sensor Data Model

8.4.1.3 Heart Rate and SPO2 level model

To determine whether a fall occurs using the heart rate and SPO2 level model, a cross-validation score was conducted to finalize the most optimal model for the dataset. The dataset was generated from real-world physiological parameters related to fall risk. The models evaluated were as follows:

Model Performance Analysis

Model	**Cross Validation Mean Accuracy**	**Model Accuracy**	**Verdict**
Logistic Regression	0.7342	0.7415	Accuracy too low, fails to detect falls reliably
Support Vector Classifier	0.8123	0.8054	Optimal but could be improved
K-Nearest Neighbors	0.8713	0.8698	Best performance, balances accuracy and generalization

Table 25: Cross Validation Assessment for Heart Rate and SPO2 Level Model

Based on the results above, the **K-Nearest Neighbors (KNN) model** was selected as the optimal model, with an accuracy close to **87%** but below **90%**, minimizing the risk of overfitting. This ensures a balance between performance and generalization when making predictions.

Hyperparameter tuning was conducted to optimize the model further, and it was determined that **7 neighbours** provided the best balance, maintaining an accuracy just under **90%** while ensuring reliable predictions.

8.4.2 Confusion Matrix

8.4.2.1 Joint Based Model

Regarding the confusion matrix for the joint based model, refer to the following figure:

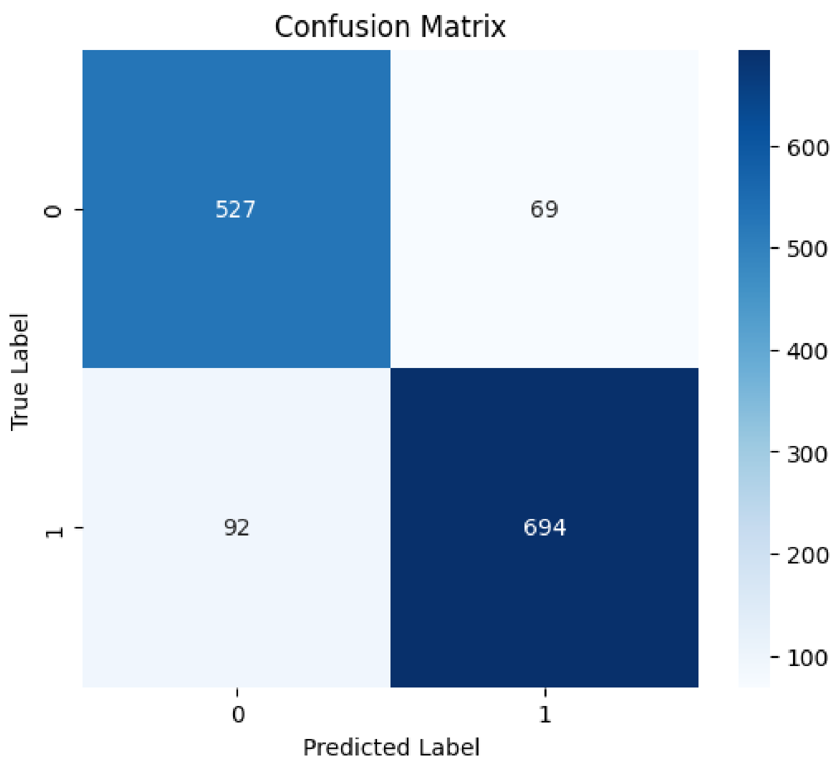


Figure 23: Confusion Matrix for Joint Based Model

The total number of predicted labels were as follows:

Metric	Description	Value
True Positives	Total number of actual falls detected	694
False Positives	Total number of false falls detected	69
True Negatives	Total number of actual non-falls detected	527
False Negatives	Total number of false non-falls detected	92

Figure 24: Confusion Matrix Metrics for joint based model

8.4.2.2 Motion Sensor Data (Accelerometer and Gyroscope) Based Model

The confusion matrix of the chosen model for the motion sensor data module (CNN) is shown below.

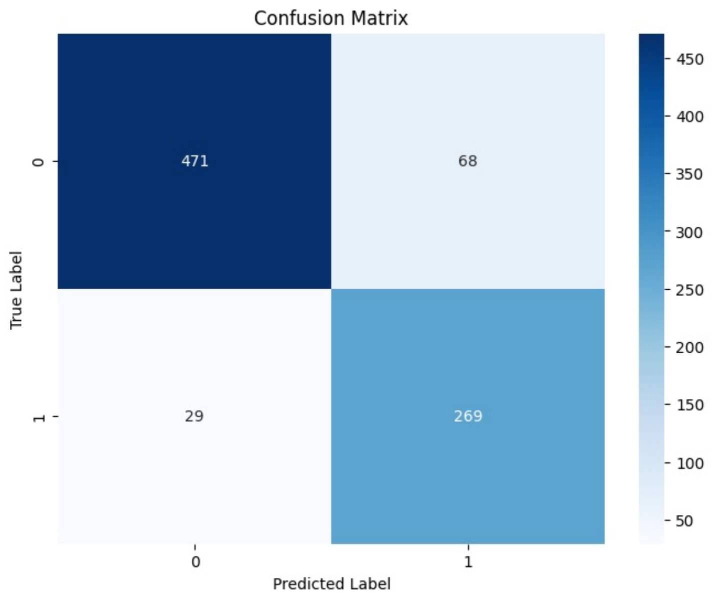


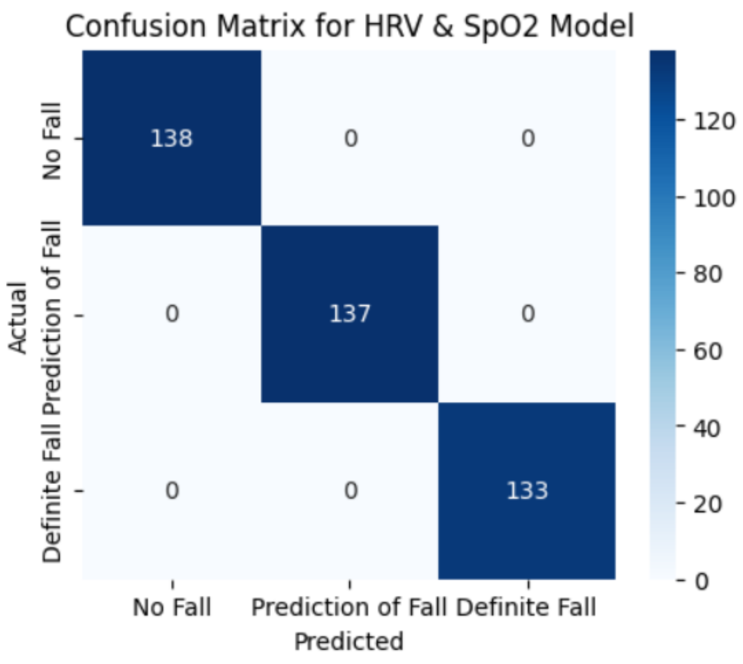
Figure 25: Confusion Matrix for Motion Sensor Data Model

Metric	Description	Value
True Positives	Total number of actual falls detected	269
False Positives	Total number of false falls detected	68
True Negatives	Total number of actual non-falls detected	471
False Negatives	Total number of false non-falls detected	29

Table 28: Confusion Matrix for Motion Sensor Data Model

8.4.2.3 Heart Rate and SPO2 level model

Regarding the confusion matrix for the HRV, SPO2 based model, refer to the following figure:



8.5 Benchmarking

The purpose of the benchmarking chapter is to compare out current product against other existing products such that the performance metrics may be assessed and evaluated to a regulated standard.

System Component	**Human Level & State-of-the-art performance**	**Comparison of the model with similar products**
Motion sensor data (Accelerometer & Gyroscope) based fall detection	Human detection of falls is often subjective and delayed, whereas these models in research achieve around 85-95% accuracy using various sensor fusion techniques or different feature extraction methods.	Unlike many traditional models that use handcrafted features and classifiers like SVM or Decision Trees, this system utilizes a CNN to process raw sensor data in its original 3D format, preserving spatial and temporal dependencies. Research studies often convert time-series data into 2D before training, whereas this approach maintains the original structure for improved accuracy.
Joint Based data (Joint projections) based fall detection model	Humans can detect a person falling based on intuition, body posture changes, behavioural changes and other visually perceptive aspects. However, their performance may be hindered by fatigue, reaction time, and availability hence limited their overall reaction time in detecting a fall. As of now there are tools and technologies available in determining if a person falls through Advanced **joint-based models** that leverage **pose estimation** (e.g., OpenPose, MediaPipe) to track skeletal joints and detect abnormal movements. It manages to achieve high accuracies (>90%), low false positives, and real-time processing on edge devices	Compared to most fall-detection models that utilize joint projection, they use conventional techniques such as pose estimation through random forests or even simple logistic regression model. However, this model supersedes them to an extend by utilizing a carefully curated dataset for both the training and test data. The data provided has joints pre-mounted onto the images that are then extracted onto a file to specify which coordinates are to be attained to categorize an active image as a fall. Furthermore, the complexity of the setup of the device is quite simple given that only a camera is required with no bodily wears for further data collection. Not to mention that the setup simply requires it that the subject to be visible from torso to toe.

HRV & SpO2-Based Fall Detection	Human assessment of fall risk based on HRV and SpO2 levels is subjective and inconsistent. AI-based models in research achieve 85-95% accuracy by leveraging deep learning and advanced time-series analysis.	Most existing systems rely on motion-based detection (accelerometers, gyroscopes) without considering biometric indicators. Our model integrates HRV and SpO2 data to predict falls before they happen, improving proactive intervention.

Table 29: Benchmarking Table

8.6 Functional Testing

Test Case	Description	Input	Expected Output	Actual Output	Remark
01	• Provide input feed from joint based model of person lying down • Provide regular heart rate and SPO2 levels • Provide simulated regular movement from user when lying down	Video Feed, Heart Rate & SPO2 Feed, gyroscopic & accelerometer feed	Assumes red-alert status but option is to deny actual fall is there on dashboard	Red-alert state achieved but fall is not approved given the fact that only one model gave an alert	PASS
02	• Provide input feed from all three models that state that the user has: • High Heart Rate • Irregular Movement • Joint Based model stating person has fallen	Video Feed, Heart Rate & SPO2 Feed, gyroscopic & accelerometer feed	Assumes red-alert and immediately alerts all saved contacts of person falling	Successfully assumes red-alert and immediately alerts all saved contacts of person falling	PASS
03	• Provide only heart rate and accelerometer/gyroscope-based feed to assess if a person falls	Heart Rate & SPO2 Feed, gyroscopic & accelerometer feed	Assumes a yellow alert if since feeds are coming from only the hear rate and movement sensors, confirmation button is provided as well	Successfully assumes yellow alert state	PASS
04	• Provide feed from each model where there are no irregular movements,	Video Feed, Heart Rate & SPO2 Feed, gyroscopic &	System should assume a green state where	System assumes a greed state successfully	PASS

	heart rates or even joint based falls	accelerometer feed	there are no errors		
05	• Disconnect all devices from the application such that the models receive no feed	Nothing	System should alert user of there being a connection issue	System receives a disconnected device error	PASS

Table 30: Functional Test Table

8.7 Module and Integration Testing

To integrate the modules and each component into the project as a single application, each component was concurrently assessed such that each threads performance was assessed as it ran alongside the other. The logic is as follows, an ETL pipeline was developed to collect data from three concurrent streams to collect the user's data, transform it to the required format and then load it into each of the models for processing. The load times and memory/machine utilization were monitored thoroughly to ensure efficiency and a regulated level of performance. To guarantee that there would be no failures experienced in the deployment of the application the following were ensured and assessed:

- Use cases were tested and tried before deploying the model to ensure that the core functionalities of the application would operate as expected
- Assess the performance of the models without having them communicate with each other to assess partial setup-based performances
- Carry out trial runs to evaluate potential issues and bugs, thereafter fixing them and making the necessary adjustments

8.8 Non-Functional Testing

8.8.1 Accuracy Testing

Accuracy testing evaluates how well the system correctly detects falls, non-falls, and risk-of-falling cases by assessing the **individual model performances** and the **final weighted voting decision**.

Each model's accuracy is tested separately before evaluating the combined system:

- **Camera-Based Fall Detection Model**: Measures how accurately it classifies body posture changes related to falls using **MediaPipe Pose and KNN classification**. Tested under varying lighting conditions and camera angles.

- **Accelerometer & Gyroscope Model**: Assesses the detection of sudden motion changes indicative of falls using data from the **MPU6050 sensor**. Tested across different movement patterns and impact scenarios.
- **Heart Rate & SpO2 Model**: Evaluates the system's ability to detect physiological anomalies (such as sudden drops in SpO2 or spikes in heart rate) that may indicate a fall risk.

The **final weighted voting model** is validated by ensuring that the **priority-based decision-making** functions as intended, with the camera model having the highest influence, followed by the accelerometer & gyroscope model, and the heart rate & SpO2 model. The accuracy of the system is determined using real-world test cases and benchmark datasets, ensuring a **high true positive rate and minimal false positives** for reliable fall detection.

8.8.2 Performance Testing

The performance of the fall detection system was evaluated by testing the individual models, as well as the combined weighted voting system. Each model's accuracy was assessed separately before evaluating the final decision-making process based on weighted voting.

8.8.2.1 Data Used for Testing:

- **Camera-Based Fall Detection Model**:
  - Dataset used: Image Based Fall Dataset
  - Number of test cases: 970
  - Number of fall cases: 208 (only utilized for training)
  - Number of non-fall cases: 208 (only utilized for training)
  - Testing conditions (e.g., lighting, angles): Fully lit task environments with full visibility of the patient/user's body
- **Accelerometer & Gyroscope Model**:
  - Dataset used: Sisfall
  - Number of test cases: 4500
  - Number of fall cases: 1798 (40%)
  - Number of non-fall cases: 2702 (60&)
  - Testing conditions (e.g., movement patterns, impact scenarios): movement patterns
- **Heart Rate & SpO2 Model**:
  - Dataset used: Cstick
  - Number of test cases: 100
  - Number of fall cases: 100
  - Number of non-fall cases: 100

8.8.3 Load Balancing

Load Balancing is a technique used to optimize resource efficiency to enhance the system responsiveness. This is done through distributing network traffic across multiple servers which helps in preventing server overload. Load balancing uses strategies such as least connections and round-robin for this purpose. This improves both system performance and reliability.

However, load balancing is not required for this system as it is hosted on a local server where network traffic is not a limiting factor.

8.9 Limitations

In terms of some of the limitations the application may experience, refer to the following:

- The joint based model can only monitor one subject at a time, so if ever multiple individuals were to be detected at a time there would be a collusion of multiple subjects
- The memory allocation and CPU utilization required by the application is beyond commercial use machines capabilities, hence making the app both difficult and expensive to run
- The overall accuracy of the system revolves around 80%, hence making the system expect false positives (and negatives) at least 20% of the time
- The recommendation system in the application is not credible enough to regard actual treatments off but may be regarded by a doctor for second opinions

8.10 Chapter Summary

In conclusion, this chapter regards an in-depth analysis into how each model was evaluated before even being trained such they perform up to a certain standard. Furthermore, the datasets referred to by each model were elaborated here as to how they were curated to fit into each model and how each model was catered to processing them with after being put through a series of hyper-parameter tuning algorithms. Furthermore, the classification reports and confusion matrixes were provided to assess the average performance per model as well as the test cases carried out to evaluate each model performance. Finally, the limitations experienced by the application were discovered and addressed in the chapter properly.

APPENDIX

References

Aziz, O. et al., 2016. A comparison of accuracy of fall detection algorithms (threshold-based vs. machine learning) using waist-mounted tri-axial accelerometer signals from a comprehensive set of falls and non-fall trials.

CDC, 2025. *Health Insurance Portability and Accountability Act of 1996 (HIPAA).* [Online] Available at: https://www.cdc.gov/phlp/php/resources/health-insurance-portability-and-accountability-act-of-1996-hipaa.html#:~:text=The%20Health%20Insurance%20Portability%20and,Rule%20to%20implement%20HIPAA%20requirements.
[Accessed 18 February 2025].

Chen, Z., Wang, Y. & Yang, W., 2021. Video Based Fall Detection Using Human Poses.

Fitriawan, H. et al., 2024. *Development of a Low-Cost Fall Detection System for the Elderly with Accurate Detection and Real-Time Alerts.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/10675589
[Accessed 14 October 2024].

Intersoft Consulting , 2024. *General Data Protection Regulation.* [Online]
Available at: https://gdpr-info.eu
[Accessed 20 February 2025].

KANDAGATLA, U. K., 2022. *Fall Detection Dataset.* [Online]
Available at: https://www.kaggle.com/datasets/uttejkumarkandagatla/fall-detection-dataset
[Accessed 10 December 2024].

Khekan, A. R., Aghdasi, H. S. & Salehpour, P., 2024. Fast and High-Precision Human Fall Detection Using Improved YOLOv8 Model.

Lazzi, A., Rziza, M. & Oulad Haj Thami, R., 2021. Fall Detection System-Based Posture-Recognition for Indoor Environments. *National Library of Medicine.*

Liaqat, S. et al., 2021. *A Hybrid Posture Detection Framework: Integrating Machine Learning and Deep Neural Networks.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/9343347
[Accessed 10 October 2024].

Lin, B. S. et al., 2022. *Fall Detection System With Artificial Intelligence-Based Edge Computing.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/9667467
[Accessed 10 October 2024].

Li, Q. et al., 2009. Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information. *University of Virginia.*

Liu, C.-P.et al., 2023. Deep Learning-based Fall Detection Algorithm Using Ensemble Model of Coarse-fine CNN and GRU Networks.

Liu, S. & Shi, C., 2024. *A Real-Time Fall Detection System Based on MoveNet and LSTM.* [Online]
Available at: https://link.springer.com/chapter/10.1007/978-3-031-70235-8_2
[Accessed 11 October 2024].

M.D, V. G. M. :. W. B. W., 2016. *Ambulatory Blood Pressure Monitoring in Older Persons.* [Online]
Available at: https://link.springer.com/chapter/10.1007/978-3-319-22771-9_11
[Accessed 11 October 2024].

Malheiros, L., Nze, G. D. A. & Cardoso, L. X., 2017. *Fall detection system and Body positioning with Heart Rate Monitoring.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/7932688/authors#authors
[Accessed 1 February 2025].

Newman-Norlund, R. D. et al., 2022. Effects of social isolation on quality of life in elderly adults. *National Library of Medicine.*

Nguyen, T.-B., Nguyen, D.-L., Nguyen , H.-Q. & Le, T.-L., 2024. *A Real-Time and Continuous Fall Detection Based on Skeleton Sequence.* [Online]
Available at: https://link.springer.com/chapter/10.1007/978-981-97-5504-2_11
[Accessed 11 October 2024].

Norsk forening for epidemiologi, 2012. *Fall risk factors in community-dwelling elderly people.* [Online]
Available at: https://oda.oslomet.no/oda-xmlui/handle/10642/1474
[Accessed 10 October 2024].

Ogundokun, R. O., Maskeliūnas, R. & Damaševičius, R., 2022. *Human Posture Detection Using Image Augmentation and Hyperparameter-Optimized Transfer Learning Algorithms.* [Online]
Available at: https://www.mdpi.com/2076-3417/12/19/10156
[Accessed 11 October 2024].

Palmerini, L., Klenk, J., Becker, C. & Chiari, L., 2020. Accelerometer-Based Fall Detection Using Machine Learning: Training and Testing on Real-World Falls.

PARLIAMENT OF THE DEMOCRATIC, 2022. *PERSONAL DATA PROTECTION.* [Online]
Available at: https://www.parliament.lk/uploads/acts/gbills/english/6242.pdf
[Accessed 20 February 2025].

QuestionPro, 2024. *Evaluation Research: Definition, Methods and Examples.* [Online]
Available at: https://www.questionpro.com/blog/evaluation-research-definition-methods-and-examples/
[Accessed 15 October 2024].

Qu, Z., Haung, T., Ji, Y. & Li, Y., 2024. Physics Sensor Based Deep Learning Fall Detection System.

RIVLIN, A. M. :. W. J. M. :. S. D. A., 1988. *Insuring Long-Term Care.* [Online]
Available at: https://connect.springerpub.com/content/sgrargg/8/1/256
[Accessed 14 October 2024].

Scribbr, 2024. *What Is a Research Design | Types, Guide & Examples.* [Online]
Available at: https://www.scribbr.com/methodology/research-design/
[Accessed 15 October 2024].

Sucerquia, A., Lopez, J. D. & Vargas-Bonilla, J. F., 2016. SisFall: A Fall and Movement Dataset. Issue Wearable Biomedical Sensors.

Tinetti, M. E., Speechly, M. & F, S., 1988. Risk Factors for Falls among Elderly Persons Living in the Community. *The New England Journal of Medicine.*

Wu, T. et al., 2019. A Mobile Cloud Collaboration Fall Detection System Based on Ensemble Learning.

Zhou, C.-C.et al., 2014. *A low-power, wireless, wrist-worn device for long time heart rate monitoring and fall detection.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/6954670
[Accessed 9 February 2025].

Zurbuchen, N., Bruegger, P. & Wilde, A., 2020. A Comparison of Machine Learning Algorithms for Fall Detection using Wearable Sensors.