Project FRIDA

Jonah Falk, Samuel Pete, Normandy Rivver, Niko Robbins, Jacob Schmoll

July 29th, 2020

Submitted to:

Dr. Razib Iqbal

Professor of Computer Science

Missouri State University

In Fulfillment of the

CSC 450 Course Project

Summer 2020

Project FRIDA		
Team Information	1. Jonah Falk 2. Samuel Pete 3. Normandy Rivver 4. Niko Robbins 5. Jacob Schmoll	
Document Type: Software Requirement Specification v1.0		*Last updated on*: 07-29-2020

Document Approval

This Software Requirements Specification has been accepted and approved by the following stakeholders:

Printed Name	Title	Signature and Date
R. IQBAL	Customer/Course Instructor	

Table of Contents

1. Introduction

This system requirements specification (SRS) will introduce and elaborate on the FRIDA software created to detect falls in an individuals place of residence. Throughout this document, the scope of the software, definitions, functionality, and any specifics needed to implement and/or run the FRIDA software will be explained in detail.

1.1 Purpose

The Purpose of this SRS is to provide a detailed overview of the FRIDA software system. This SRS will describe the product, its functionality and user interaction that the middleware can provide. The document is intended for Dr. Iqbal, Snigdha Chaudhari, Shashi Khanal, any future stakeholders, the current development team, and any future students that may work on a related project.

1.2 Scope

The FRIDA system will serve as a preleminary step in developing a software system capable of being used inside a person's place of residence to detect falls. To achieve this goal, the software will be designed to detect falls at an accuracy rate of greater than or equal to 80% through real-time tracking of an individual inside their residence to montior for fall sitautions. The software will not be designed to track multiple individuals at once. Notification of when a fall is occuring will be issued after the system has detected a fall. Overall, the major benefit of the FRIDA system will be confirmation in the form of an minimal viable product for detection of falls in those needing protection and montioring for fall situations.

1.3 Definitions, Acronyms, and Abbreviations

CNN – Convolutional Neural Network
FRIDA – Fall Rate Identification, Detection, and Analysis
IDE – Integrated Development Environment
kbps – kilobits per second
ONNX – Open Neural Network Exchange
rad – radians
User - Someone who the FRIDA system is tracking for potential falls.

1.4 References

A. Núñez-Marcos, G. Azkune, and I. Arganda-Carreras, "Vision-Based Fall Detection with Convolutional Neural Networks," *Hindawi*, vol. 2017, Dec. 2017.

K. Singh, A. Rajput, S. Sharma, "Fall Detection Using Machine Learning Methods: A Survey," *International Journal of Mathematical, Engineering and Management Sciences*, August 2019.

S. D. Hernandez, Y. DeLaHoz and M. Labrador, "Dynamic background subtraction for fall detection system using a 2D camera," *2014 IEEE Latin-America Conference on Communications (LATINCOM)*, Cartagena de Indias, 2014, pp. 1-6.

W. Chen, Z. Jiang, H. Guo, and X. Ni, “Fall Detection Based on Key Points of Human-Skeleton Using OpenPose,” *MDPI*, May 2020.

1.5 Overview

The following SRS contains a description of the product, its functions, and other factors that will affect the product. Following that section, the requirements will be covered, including both functional and non-functional. This will also describe design constraints, interface requirements, and hardware requirements.

2. General Description

This section provides an overview of the FRIDA system with an emphasis on explainations to show how it will work and to introduce the basic functionality of the software. It will also describe how users will use the system and what functionalities are available. Lastly, the constraints, assumptions, and dependencies for the software will be presented.

2.1 Product Perspective

There are many products on the market to help seniors with fall assistance. One of the most known is Life Alert. Life Alert is a portable device with a button that a senior citizen can wear and press after a fall, of which then alerts medical emergency services and/or family members. Other product examples include Bay Alarm Medical, Mobile Help, and Life Station. Life Station in particular does have fall detection in its wristband, but it is not the main signal of help.

The FRIDA software's ultimate goal is to specifically specialize in detecting whether or not an elderly user needs help from a fall via outputting a fall detection notification onto a computer. This is the most common difference between the FRIDA project and other fall assistance systems.

2.2 Product Functions

User Detection: Tracks the user for potential falls.
Fall Detection: Detects when the user completely falls down to the point they stay on the ground at a predicted rate of 80% or more via utilizing a CNN.
Notification: Outputs continuous fall and non-fall probability prediction percentages to the system's console and then a fall detection notification when a fall occurs. Additionally it outputs system loading, error, success, and termination statements.

2.3 User Characteristics

The main demographic of the FRIDA project will be senior citizens, most likely over the age of 65 who reside in a nursing home facility. Physically disabled users would also be able to use this, as well as users with injuries that affect their balance.

With the main demographic being 65+, these users will be more prone to injury. They will also be slow-moving, so the user tracking would not be required to be as fast as it would if it were to track a younger person or child.

One problem with the demographic is that the user must have an Internet connection in their household, which is not always common for senior citizens.

Developers will also be a primary demographic since some aspects related to development of the FRIDA project will require specific user needs to be meant for developers to provide a functional, maintainable, performance-based, dependable, and usable software system.

2.4 General Constraints

While the budget is not anticipated to be an issued, one constraint would be the fact that if the open source OpenCV library does not work for the project's needs, there would a necessity of having to look into other libraries, of which many often come with a cost. For instance, WrnchAI is a real-time computer vision library that can cost up to $6,000 annually with a $500 monthly subscription. Since the project's funding is $0, in the instance the OpenCV library is inadequate for the project's needs, we would need develop the necessary code that would suffice for the project.

Internet connection is a constraint for the software because its setup process includes downloading programs and libraries from online sources. Due to this, a stable internet connection is needed for the software to function properly.

The location of the camera is a constraint because the live video feed tracking needs to occur indoors since the user's webcam is not expected to be weatherproof. Moreover, it needs to be physically located where eldery people are most likely to be, such as their residence, theirs family's, or a nursing home.

The most crucial constraint is time. The final deadline is 07/29/20, so there is approximately 8 weeks to ensure that the software's implementation is completed, which includes training the system's model thoroughly enough via machine learning as to achieve an 80% accuracy fall detection rate.

2.5 Assumptions and Dependencies

The usage of the OpenCV library and other supporting libraries is dependent on each developer's individual computer having adequate processing power. The OpenCV library will be capable of providing the overall framework for fall detection, and additional libraries, namely ONNX and PyTorch, will help with the utilization of a CNN to detect falls, and then to capture and analyze corresponding data. The brightness of the room where the webcam will be physically located will be bright enough to see the outline of the of body (i.e., silhouette) of the person being tracked. There will be at most one person in the room where the webcam will be physically located. The webcam's viewing angle will be positioned downward between 315 degrees ($\frac{5\pi}{3}$ rad) and 0 degrees (2π rad).

3. Specific Requirements

3.1 Functional Requirements

3.1.1 The system must detect a webcam.

Source: Product Description
Priority: 1
Introduction: The FRIDA software must be able to detect a webcam.
Inputs: Webcam access
Processing: None
Outputs: None
Error Handling: If no webcam is detected, the program shall output an error message to the system console and then close.
Related Requirements: None

3.1.2 The system must connect to the webcam.

Source: Product Description
Priority: 1
Introduction: The FRIDA software must be able to connect to a webcam.
Inputs: Webcam connection
Processing: None
Outputs: None
Error Handling: If no webcam connection can be established, the program shall output an error message to the system console and then close.
Related Requirements: None

3.1.3 The system must be able to access live video from a webcam.

Source: Product Description
Priority: 1
Introduction: The FRIDA software will receive and capture live video input from a webcam.
Inputs: Live video feed
Processing: The streaming video will be captured via OpenCV using Python.
Outputs: None
Error Handling: If no video is detected, the program shall output an error message to the system console and then close.
Related Requirements: None

3.1.4 The system must be able to detect a human when using a live video feed.

Source: Product Description
Priority: 2
Introduction: The FRIDA software shall analyze the captured live video feed to determine the presence of a human user in the video.
Inputs: Live video feed

Processing: The live video feed shall be analyzed by using OpenCV and a CNN via ONNX, PyCharm, and other supporting libraries to determine the presence of a human.
Outputs: None
Error Handling: None
Related Requirements: None

~~3.1.5 The system must put a rectangle box around the person under review so that developers can ensure the object tracking and object detection is working.~~
~~Source: Product Description~~
~~Priority: 2~~
~~Introduction: Upon detecting a human presence, the FRIDA software shall put a rectangle box around them.~~
~~Inputs: Video~~
~~Processing: The video shall be analyzed via OpenCV using Python to calculate the box's position in an Cartesian coordinate system with a determined width and height for the 4 points in the rectangle.~~
~~Outputs: 4 points of the corners of the rectangle in an Cartesian coordinate system.~~
~~Error Handling: If no rectangle is able to be formed due to an issue, the system will output a warning of stating it is unable to form a rectangle around the person under review, but it will continue trying to do so.~~
~~Related Requirements: 3.2.3, 3.2.4, 3.2.10~~

3.1.6 The system shall print a message to the system console when a fall is detected when using a live video feed.
Source: Team Discussion
Priority: 4
Introduction: The FRIDA software shall print a message to the system console consisting of "FALL DETECTED".
Inputs: Fall event
Processing: The system shall print a message to the system console.
Outputs: Console message (text)
Error Handling: None
Related Requirements: 3.2.1, 3.2.2

~~3.1.7 The system must be able to function in the cloud.~~
~~Source: Product Description~~
~~Priority: 1~~
~~Introduction: To ensure the system is capable of operating in the cloud, band consumption rate should be minimized between 384 kbps and 500 kbps.~~
~~Inputs: Streaming video~~
~~Processing: None~~
~~Outputs: None~~
~~Error Handling: None~~
~~Related Requirements: 3.2.8~~

~~**3.1.8 The system must be able to determine the initial position of the person under review when using a live video feed.**~~

~~Source: Product Description~~
~~Priority: 1~~
~~Introduction: Upon startup, the system must be able to determine the initial position of a human so that an initial state can be determined.~~
~~Inputs: Initial human position data~~
~~Processing: The live video feed shall be analyzed by using OpenCV to determine whether the position of a human is in a fall or non-fall (e.g., standing, sitting, etc.) state.~~
~~Outputs: If a fall has taken place, the FRIDA software will output a fall status notification to the system console. Otherwise, it will output a non-fall status notification.~~
~~Error Handling: None~~
~~Related Requirements: 3.2.3, 3.2.4, 3.2.5, 3.2.6, 3.2.7, 3.2.10~~

3.1.9 The system must be able to determine the initial state is a fall state of the person under review when using a live video feed.

Source: Product Description
Priority: 1
Introduction: The initial state of the person under review must be determined.
Inputs: Human position data via live video feed
Processing: If the human state data passes the given thresholds, the FRIDA software will determine that the initial state is a fall state.
Outputs: If a fall has taken place, the FRIDA software will output a fall statement to the console.
Error Handling: If an initial state cannot be determined, the system will conintue to monitor.
Related Requirements: None

3.1.10 The system must determine if a current state is a fall state when using a live video feed.

Source: Product Description
Priority: 1
Introduction: The system must determine if a current state is a fall state. The current state is defined as the state after the initial state.
Inputs: Human position data via a live video feed
Processing: If the human position data passes the given thresholds, the FRIDA software will determine that the current state is a fall state. Otherwise, the current state will be a non-fall state.
Outputs: If a fall has taken place, the FRIDA software will output a fall notification to the system console.
Error Handling: None
Related Requirements: None

3.1.11 The system must determine if the current state is a fall state when using a video dataset file.

Source: Product Description
Priority: 1

Introduction: The system must determine if the current state is a fall state. The current state is defined as the state after the initial state.
Inputs: Human position data via a video dataset file
Processing: If the human position data passes the given thresholds, the FRIDA software will determine that the current state is a fall state. Otherwise, the current state will be a non-fall state.
Outputs: If a fall has taken place, the FRIDA software will output a fall notification to the system console.
Error Handling: None
Related Requirements: None

3.1.12 The system must determine if current state of a human is a movement state when using a live video feed.

Source: Product Description
Priority: 1
Introduction: The system needs to determine if a given current state is a movement state (i.e., is the person moving).
Inputs: Human position data via live video feed
Processing: If the human position data passes the given thresholds, the FRIDA software will determine the current state is a moving state.
Outputs: If movement occurs, the FRIDA software will track the human.
Error Handling: None
Related Requirements: None

3.1.13 The system must determine if initial state is a non-fall state when using a live video feed.

Source: Product Description
Priority: 1
Introduction: The system must determine an initial non-fall state (e.g., standing, sitting, etc.), if such a state is occuring, upon startup.
Inputs: Human position data via live video feed
Processing: If the human position data passes the given thresholds, the FRIDA software will determine that the initial state as a non-fall state.
Outputs: The system will continue to monitor for a fall.
Error Handling: None
Related Requirements: None

3.1.14 The system must determine if an initial state is a non-fall state when using a video dataset file.

Source: Product Description
Priority: 1
Introduction: The system must determine an initial non-fall state (e.g., standing, sitting, etc.), if such a state is occuring, upon startup.
Inputs: Human position data via a video dataset file
Processing: If the human position data passes the given thresholds, the FRIDA software will determine that the initial state as a non-fall state.

Outputs: The system will continue to monitor for a fall.
Error Handling: None
Related Requirements: None

3.1.15 The system must determine if an initial state is a moving state when using a live video feed.

Source: Product Description
Priority: 1
Introduction: The system must determine an initial moving state, if such a state is occuring, upon startup.
Inputs: Human position data via live video feed
Processing: If the human position data passes the given thresholds, the FRIDA software will determine that the initial state as a moving state.
Outputs: If a movement state is occuring, the FRIDA software will track the human.
Error Handling: None
Related Requirements: None

3.1.16 The system must track a human in a moving state when using a live video feed.

Source: Product Description
Priority: 1
Introduction: While in a human is in a moving state, the system must track the movement of the human.
Inputs: Human position data via live video feed
Processing: If the human position data passes the given thresholds, the FRIDA software will track the human in a moving state.
Outputs: If a movement state is occuring, the FRIDA software will track the human.
Error Handling: None
Related Requirements: None

~~3.1.17 The system must determine an updated state of a human after their movement state ends when using a live video feed.~~

~~Source: Product Description~~
~~Priority: 1~~
~~Introduction: After a human stops moving, the system must update the state of the human.~~
~~Inputs: Human position data~~
~~Processing: If the human position data passes the given thresholds, the FRIDA software will update the state after the movement state ends.~~
~~Outputs: The updated the state of a person is printed to the system console.~~
~~Error Handling: None~~
~~Related Requirements: 3.2.3, 3.2.4, 3.2.5, 3.2.6, 3.2.7, 3.2.10~~

~~3.1.18 The system must determine the updated state of a human after their movement state ends when using a video dataset file.~~

~~Source: Product Description~~
~~Priority: 1~~

~~Introduction: After a human stops moving, the system must update the state of the human.~~
~~Inputs: Human position data~~
~~Processing: If the human position data passes the given thresholds, the FRIDA software will update the state after the movement state ends.~~
~~Outputs: The updated the state of a person is printed to the system console.~~
~~Error Handling: None~~
~~Related Requirements: 3.2.3, 3.2.4, 3.2.5, 3.2.6, 3.2.7, 3.2.10~~

~~3.1.19 The system must provide a notification when it is no longer able to monitor the person at risk for fall.~~

~~Source: Product Description~~
~~Priority: 1~~
~~Introduction: When the person under review is no longer being monitored by the system, a notification must be provided to inform staff.~~
~~Inputs: Human position data~~
~~Processing: If the human position data is unable to be obtained within 3 seconds, the FRIDA software will alert the user and conintue to detect the person requested for protection.~~
~~Outputs: Warning message~~
~~Error Handling: None~~
~~Related Requirements: 3.2.3, 3.2.4, 3.2.5, 3.2.6, 3.2.7~~

3.1.20 The system must be able to access video from a dataset file.

Source: Product Description
Priority: 1
Introduction: The FRIDA software will receive and capture video input from a dataset file.
Inputs: Video dataset file
Processing: The streaming video will be captured via OpenCV using Python.
Outputs: None
Error Handling: If no video is detected, the program shall output an error message to the system console and then close.
Related Requirements: None

3.1.21 The system must be able to detect a human when using a video dataset file.

Source: Product Description
Priority: 2
Introduction: The FRIDA software shall analyze the captured video dataset file input to determine the presence of a human user in the video.
Inputs: Video dataset file
Processing: The video dataset file shall be analyzed by using OpenCV and a CNN via ONNX, PyCharm, and other supporting libraries to determine the presence of a human.
Outputs: None
Error Handling: None
Related Requirements: None

3.1.22 The system shall print a message to the system console when a fall is detected when using a video dataset file.

Source: Team Discussion
Priority: 4
Introduction: The FRIDA software shall print a message to the system console consisting of "FALL DETECTED".
Inputs: Fall event
Processing: The system shall print a message to the system console.
Outputs: Console message (text)
Error Handling: None
Related Requirements: 3.2.1, 3.2.2

3.1.23 The system must be able to determine the initial state is a fall state of the person under review when using a video dataset file.

Source: Product Description
Priority: 1
Introduction: The initial state of the person under review must be determined.
Inputs: Human position data via a video dataset file
Processing: If the human state data passes the given thresholds, the FRIDA software will determine that the initial state is a fall state.
Outputs: If a fall has taken place, the FRIDA software will output a fall event.
Error Handling: If an initial state cannot be determined, the system will conintue to monitor.
Related Requirements: None

3.1.24 The system must determine if current state of a human is a movement state when using a video dataset file.

Source: Product Description
Priority: 1
Introduction: The system needs to determine if a given current state is a movement state (i.e., is the person moving).
Inputs: Human position data via a video dataset file
Processing: If the human position data passes the given thresholds, the FRIDA software will determine the current state is a moving state.
Outputs: If movement occurs, the FRIDA software will track the human.
Error Handling: None
Related Requirements: None

3.1.25 The system must determine if an initial state is a moving state when using a video dataset file.

Source: Product Description
Priority: 1
Introduction: The system must determine an initial moving state, if such a state is occuring, upon startup.
Inputs: Human position data via video dataset file

Processing: If the human position data passes the given thresholds, the FRIDA software will determine that the initial state as a moving state.
Outputs: If a movement state is occuring, the FRIDA software will track the human.
Error Handling: None
Related Requirements: None

3.1.26 The system must track a human in a moving state when using a video dataset file.

Source: Product Description
Priority: 1
Introduction: While in a human is in a moving state, the system must track the movement of the human.
Inputs: Human position data via a video dataset file
Processing: If the human position data passes the given thresholds, the FRIDA software will track the human in a moving state.
Outputs: If a movement state is occuring, the FRIDA software will track the human.
Error Handling: None
Related Requirements: None

3.2 Non-Functional Requirements

3.2.1 The system must detect a fall at least 80% of the time when using a live video feed.

Source: Customer Meeting
Priority: 3
Introduction: Under test conditions, the FRIDA software must successfully output a fall event at least 80% of the time.
Inputs: Human position data via live video feed
Processing: See 3.1.4
Outputs: A fall notification is printed to the system console.
Error Handling: None
Related Requirements: 3.1.4, 3.2.2

3.2.2 The system must detect a fall within 10 seconds of it occurring when using a live video feed.

Source: Customer Meeting
Priority: 3
Introduction: From the time that the fall occurs to the time that FRIDA outputs a fall event, no more than 10 seconds must pass.
Inputs: Live video feed
Processing: See 3.1.4
Outputs: A fall notification is printed to the system console.
Error Handling: If the system takes longer than 10 seconds to output a fall event, it shall be treated as if the system did not recognize the fall per 3.2.1.
Related Requirements: 3.1.4, 3.2.1

~~3.2.3 The system must recognize the presence of a human 100% of the time.~~

~~Source: Team Discussion~~
~~Priority: 2~~

~~Introduction: The FRIDA software must determine the presence of a human subject in at least 100% of the frames in which a human subject actually appears.~~
~~Inputs: Live video feed~~
~~Processing: Testing this requirement shall consist of giving the system's human recognition system individual images to determine success rate.~~
~~Outputs: Human position data to the system console, if a human is detected.~~

~~3.2.4 The system must recognize the position of a human 100% of the time.~~
~~Source: Team Discussion~~
~~Priority: 2~~
~~Introduction: The FRIDA software must determine the position of a human subject in at least 100% of the frames in which a human subject actually appears.~~
~~Inputs: Live video feed~~
~~Processing: Testing this requirement shall consist of giving the system's human individual images to determine position which will allow for a determination success rate.~~
~~Outputs: Human position data to the system console, if a human is detected.~~
~~Error Handling: None~~
~~Related Requirements: 3.1.2~~

~~3.2.5 The system must recognize the state of a human 100% of the time.~~
~~Source: Team Discussion~~
~~Priority: 2~~
~~Introduction: The FRIDA software must recognize the state of a human subject in at least 100% of the frames in which a human subject actually appears.~~
~~Inputs: Image~~
~~Processing: Testing this requirement shall consist of giving the system's human state situations using individual images to determine success rate.~~
~~Outputs: Human state data to the system console, if a human is detected and position is determined.~~
~~Error Handling: None~~
~~Related Requirements: 3.1.2~~

~~3.2.6 The system must differentiate between a fall state and similar states, such as sleeping.~~
~~Source: Team Discussion~~
~~Priority: 2~~
~~Introduction: The FRIDA software must ensure bandwidth consumption is between 384 kbps and 500 kbps to minimize costs to the users.~~
~~Inputs: Live video feed~~
~~Processing: Uploading video data to cloud provider~~
~~Outputs: None~~
~~Error Handling: None~~
~~Related Requirements: 3.1.4, 3.1.9, 3.1.10, 3.1.11, 3.1.12, 3.1.14, 3.1.15, 3.1.17, 3.1.18~~

~~3.2.7 The system must differentiate between a moving state and states, such as standing and sitting.~~
~~Source: Team Discussion~~

~~Priority: 2~~
~~Introduction: The FRIDA software must ensure bandwidth consumption is between 384 kbps and 500 kbps to minimize costs to the users.~~
~~Inputs: Live video feed~~
~~Processing: State determination~~
~~Outputs: Current state~~
~~Error Handling: None~~
~~Related Requirements: 3.1.4, 3.1.7, 3.1.9, 3.1.10, 3.1.13, 3.1.16, 3.1.17, 3.1.18~~

~~3.2.8 The system must minimize cloud bandwidth consumption rate between 384 kbps and 500 kbps to minimize costs to the users.~~
~~Source: Team Discussion~~
~~Priority: 2~~
~~Introduction: The FRIDA software must ensure bandwidth consumption is between 384 kbps and 500 kbps to minimize costs to the users.~~
~~Inputs: Video feed~~
~~Processing: Uploading video data to cloud provider~~
~~Outputs: NA~~
~~Error Handling: None~~
~~Related Requirements: 3.1.7~~

~~3.2.9 The system must provide a warning within 3 seconds of object detection loss.~~
~~Source: Customer Meeting~~
~~Priority: 3~~
~~Introduction: From the time that the object detection loss occurs to the time that FRIDA outputs a warning event, no more than 3 seconds must pass.~~
~~Inputs: Live video feed~~
~~Processing: See 3.1.4~~
~~Outputs: A warning message event is printed to the system console.~~
~~Error Handling: None~~
~~Related Requirements: 3.1.4, 3.1.5, 3.2.1~~

~~3.2.10 The system must be able to track at most 1 individual in a moving state.~~
~~Source: Product Description~~
~~Priority: 1~~
~~Introduction: The FRIDA software must be able to track at most 1 individual moving through their enivronment if they are in camera view.~~
~~Inputs: Streaming video~~
~~Processing: The streaming video will be captured via OpenCV using Python.~~
~~Outputs: None~~
~~Error Handling: If unable to track, a warning must be provided and the program must continue running in attempt to regain tracking of person under review.~~
~~Related Requirements: 3.2.3, 3.2.4, 3.2.5, 3.2.7~~

3.2.11 The system must detect a fall at least 80% of the time when using a video dataset file.
Source: Customer Meeting
Priority: 3
Introduction: Under test conditions, the FRIDA software must successfully output a fall event at least 80% of the time.
Inputs: Human position data via a video dataset file
Processing: See 3.1.4
Outputs: A fall notification is printed to the system console.
Error Handling: None
Related Requirements: 3.1.4, 3.2.2

3.2.12 The system must detect a fall within 10 seconds of it occurring when using a video data file.
Source: Customer Meeting
Priority: 3
Introduction: From the time that the fall occurs to the time that FRIDA outputs a fall event, no more than 10 seconds must pass.
Inputs: Video dataset file
Processing: See 3.1.4
Outputs: A fall notification is printed to the system console.
Error Handling: If the system takes longer than 10 seconds to output a fall event, it shall be treated as if the system did not recognize the fall per 3.2.1.
Related Requirements: 3.1.4, 3.2.1

3.2.13 The system must detect between a fall state and non-fall state.
Source: Customer Meeting
Priority: 3
Introduction: Given a users state, the system must differentinate between the state as a fall state or non-fall state..
Inputs: Video dataset file or live video feed.
Processing: See 3.1.4
Outputs: A fall notification is printed to the system console when a fall is detected. Otherwise, the system continues to monitor for a fall.
Error Handling: None
Related Requirements: 3.1.1, 3.1.2, 3.1.3, 3.1.4, 3.1.9, 3.1.10, 3.1.11, 3.1.12, 3.1.13, 3.1.14, 3.1.15, 3.1.16, 3.1.20, 3.1.21, 3.1.23, 3.1.24, 3.1.25, 3.1.26

3.3 Design Constraints

1) The user must have Windows 10 64-bit with administrator privileges on their computer.
2) Usage of the Anaconda distribution and Python programming language, alongside basic understanding on how to comment and uncomment lines of code.
3) Usage of the PyCharm IDE, of which the system must be operated within it.
4) Usage of the OpenCV and NumPy libraries.
5) Usage of the ONNX and ONNX Runtime libraries, alongside their included supporting libraries.

6) Usage of the PyTorch and scikit-learn libraries, alongside their included supporting libraries.
7) If using the live video feed option, the webcam must not have an obstructed view of the individual under review.

3.4 External Interface Requirements

3.4.1 User Interfaces

UI.1 The FRIDA software must be launched from within the PyCharm IDE.

Source: Customer Meeting
Priority: 1
Introduction: The user launches the program from within the PyCharm IDE.
Inputs: IDE interaction
Processing: The program will connect and launch the video frame.
Outputs: Shows either the live video feed from a webcam or the video from the dataset file, and output data into the IDE's console.

UI.2 The FRIDA software must launch at least 1 video frame.

Source: Team Meeting
Priority: 1
Introduction: The user launches the program from within the PyCharm IDE, which loads 1 or 2 video frame(s).
Inputs: IDE interaction
Processing: The user will choose to use either the grayscale or background subtraction video frame outputs, or both. Grayscale will be the default option. Afterward, the program will connect and launch the video frame(s).
Outputs: Shows either the live video feed from a webcam or the video from the dataset file in grayscale, background subtraction, or both.

3.4.2 Hardware Interfaces

Not Applicable.

3.4.3 Software Interfaces

SIR.1 The user must have the required libraries to run.

Source: Customer Meeting
Priority: 2
Introduction: The FRIDA software requires the OpenCV, NumPy, ONNX, ONNX Runtime, PyTorch, scikit-learn, and their included supporting libaries to run.
Inputs: Installation via the Anaconda distribution and through PyCharm's built-in terminal.
Processing: OpenCV to capture images from the webcam; NumPy for array and mathematical functions; and ONNX, ONNX Runtime, PyTorch, and sckit-learn for utilizing a CNN.
Outputs: None
Error Handling: Print a error to Pycharm's console stating that a specific library is missing.

SIR.2 The user must have the Anaconda distribution.

Source: Customer Meeting

Priority: 2
Introduction: The FRIDA software requires the Anaconda distribution to install the NumPy, PyTorch, scikit-learn libraries, and their included supporting libraries.
Inputs: The Anaconda interface.
Processing: Create a virtual environment, import the libraries, and launch PyCharm from within the distribution.
Outputs: None
Error Handling: None

SIR.3 The user must have the PyCharm IDE.
Source: Customer Meeting
Priority: 2
Introduction: The FRIDA software requires the PyCharm IDE to install the OpenCV, ONNX, ONNX Runtime libraries, and their included supporting libraries.
Inputs: PyCharm's interface and terminal.
Processing: Create a Python interpreter using Anaconda's virtual environment and install the remaining libraries in PyCharm's terminal.
Outputs: The display of the chosen Python interpreter and, upon installation of the remaining libraries, success and/or error statements in PyCharm's terminal.
Error Handling: None

3.4.4 Communications Interfaces

CIR.1 The FRIDA software must output data into PyCharm's console.
Source: Team Meeting
Priority: 1
Introduction: The user launches the program from within the PyCharm IDE.
Inputs: IDE interaction
Processing: The software will analyze if the program has initiated, and then if so, analyze if a webcam or video dataset file was detected. The software will then collect and analyze fall data. It will also determine if the program was terminated at any point.
Outputs: A loading statement will be printed to the console when the program is initiated, and it will either output an error or success statement if the video frame was able to be loaded. When the video frame is running, non-fall and fall prediction percentages will be continuously printed. If a fall is detected, it will print a fall detection statement. If will print a termination statement if a live video feed frame is terminated by the user by press 'q' on their keyboard.

CIR.2 The FRIDA software must output fall status data onto a live video feed frame.
Priority: 2
Introduction: The FRIDA software will display a heads-up display that state states whether the system has actively detected a fall or not.
Processing: The software will analyze if a fall has been detected.
Outputs: A message within the live video feed frame stating "Status: FALL DETECTED" in a red font if a fall is detected, and then otherwise if not, it states "Status: Idle" in a green font.
Error Handling: None

CIR.3 The FRIDA software must accept a keyboard input to terminate the program when using the live video feed option.
Source: Team Discussion
Priority: 2
Introduction: The FRIDA software will accept input from the keyboard to end camera capture and close the program.
Processing: Recognize an input of ‘q’ from the keyboard and close the program.
Outputs: A message informing the user that the software was terminated.
Error Handling: None

3.5 Logical Database Requirements

The FRIDA system utilizes OpenCV for the program’s framework in that it will allow the software to create video frames and display a heads-up display with the fall detection status within a live video feed frame, alongside creating the overall foundation of utilizing other libraries, such as ONNX and PyTorch, to implement a CNN for fall detection, identification, and analysis. These libraries will allow an option to test fall detection in video dataset files as well. Therefore, OpenCV, ONNX, PyTorch, and other supporting libraries (e.g., ONNX Runtime, NumPy, scikit-learn, etc.) will be required to be installed for the program to run. Moreover, the program will require the Anaconda distribution and PyCharm’s built-in terminal to install these libraries, and it will need specifically the PyCharm IDE to run the program itself.

3.6 Other Requirements

Not Applicable.

4. Change Management Process

Any changes to the FRIDA project should be able to be categorized within 1 of the 3 given following situations. Definitions are reusable and have an explanation to either introduce the concept or update the concept within the specific situation.

1. Developer-Requested Change – One of the developers sees an opportunity for a change for the project at any point.
   - Discord: The developer that comes up with the change should communicate with the rest of the team within 24 hours that they have an idea for a change. They should state a broad overview of the suggested change so that the other team members can understand why the change is being suggested.
   - Zoom: At least 3 of the 5 team members should meet via Zoom within 24 hours of the notification of the change and thoroughly discuss the change being suggested. Pros and cons should be considered, such as performance, setbacks, side effects, and time management. In the event the requested change will take more than a 2 weeks to complete, we will need to communicate clearly with the stakeholders about how to proceed with the situation. At the end of the Zoom meeting, which should be at most 1 hour, there should be no ambiguity about the change between any of the attending team members.
   - Vote: Within 24 hours after the discussion on Zoom, the team should vote on whether or not to implement the proposed change. There must be a minimum of 3 votes for the implementing the change in order to move forward with it.
   - Report: The team should formally type up the change that states why the team feels like the change is needed, along with any other valuable information needed to explain the change fully, within 24 hours after the vote.
   - Stakeholders: Within 24 hours after the report is typed up, the team will present the formal report to the stakeholders up to ensure that both the team and stakeholders agree on the change. Discussion with the stakeholders can be done at the this time via Zoom and/or email.

2. Stakeholder-Requested Change – One of the stakeholders determines a change in the project is needed at any point and is brought to the developer team.
   - Zoom
   - Report
   - Stakeholders: The team will present back to the stakeholders within 24 hours what they feel like the change will do to the project and bring up any setbacks the stakeholders might not have been aware of when first determining the requested change.

Appendices

Not Applicable.