Software Requirements Specification

for

YOLO

Version 1.1

Prepared by
Muhammed Haseeb MOTAN
2398071
Tayyip ÖZTÜRK
2380806

Date: 18 April 2022

Table of Contents

List of Figures

List of Tables

1 Introduction

This document encapsulates the Software Requirements Specification (SRS) of YOLO (Your Own Living Object), which is an autonomous robotic toy made for enhancing the creativity of children and their parents during their free play activities. The official document for YOLO can be accessed from here[1]:

1.1 Purpose of the System

The purpose of this project is creating new story-lines to users of YOLO, which are ordinarily children, their parents and developers of the project. In this wise, children have the opportunity of developing their social and analytical skills via interacting a social robot and experiencing different behaviors of robot during their free play activities.

1.2 Scope

The target audience of this project is primarily children. In a manner suitable for the purposes of the system, YOLO can be used in various environments and for different purposes:

- Education
- Free play activities
- Physical and/or mental rehabilitation of children

The system has been designed and developed in accordance with the capabilities that it be needed to serve in these fields and purposes. As a system, YOLO (Your Own Living Object) includes different subsystems and parts, which enables it to interact with user in a manner of social interaction. Thus it has been equipped with a compatible layers of hardware including:

- Raspberry-Pi W Zero
- Optical sensor and touch sensor
- Motors, their drivers and dockings
- Wheels and wheels layer
- Batteries and their layer

[1] https://www.researchgate.net/publication/340367036˙YOLO˙-˙Your˙Own˙Living˙Object

- Controller boards and their layer
- Glowing fibers layer and their layer
- A shell that envelops the hardware system of the system and protects it against damage.

Those components also gives the ability to the system interact with user by combining movements, light states and animations. Moreover, the system is equipped with a software that infers user's behaviour pattern and act differently for each user, besides developing YOLO over the API provided and creating new behaviour profiles manually.
From stakeholder's point of view, YOLO is designed and developed to be used by various organizations and communities, such as:

- Educational Institutions
- Healthcare Institutions
- Families

Although YOLO approximately has a $200 expense of development per unit and can be considered expensive, it brings many benefits for these communities, however, a medical treatment including physical and mental rehabilitation of a child can not be measured by money. Moreover, this expenditure is nothing compared to its value and compared to the expenditure of other techniques used by stakeholders to get similar results.

1.3 System Overview

1.3.1 System Perspective

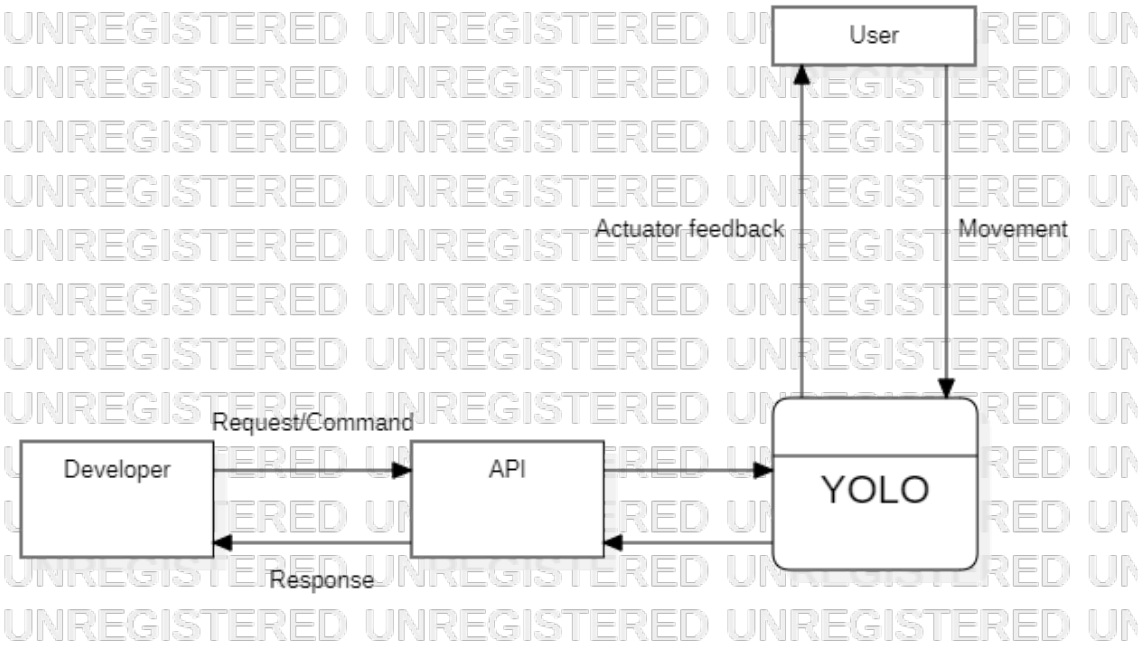

Figure 1.1: System Context Diagram

The system context diagram of YOLO indicates the external entities interacting with the system, and their relations with it. Over these relations, YOLO can fulfill its objectives. Entities are shown in the Figure 1.1 above.

1.3.1.1 System Interfaces

- User Interfaces
- Hardware Interfaces
- Software Interfaces
- Communication Interfaces
- Memory

1.3.1.2 User Interfaces

Although YOLO does not provide any software application for user to interact due to its main purpose, it is equipped with touch sensors to serve user a physical interface.

Figure 1.2: Physical Interaction of a Child with YOLO[2]

1.3.1.3 Hardware Interfaces

YOLO has touch sensors to recognize physical contacts and optical sensor to detect the movement direction. Via these sensors, YOLO recognizes, identifies and categorizes each movement.[2]
YOLO has wheel actuators to provide itself navigation. These actuators can be active due to a profile created by developer or a profile YOLO inferred from user's behaviours. Also LED actuators can be used in these stages to indicate states for user to understand, by a profile created by a developer or a profile inferred by YOLO itself.

1.3.1.4 Software Interfaces

YOLO has a Raspberry-Pi W Zero inside. This system uses Python as a development language, over the operating environment Raspbian Stretch Lite OS.

1.3.1.5 Communication Interfaces

YOLO has an wireless connectivity ability thanks to Raspberry-pi inside it. Over a wireless Wi-Fi router, a developer can establish a connection with YOLO over the API. After connection

[2]https://www.researchgate.net/publication/340911329˙Software˙architecture˙for˙YOLO˙a˙creativity-stimulating˙robot

is established successfully, communication including sending commands that sets a new profile is available.

1.3.1.6 Memory Constraints

YOLO consists an SDHC SD card to store required data. Since there is no data coming from any camera or voice recorder, the data stored is related to its state and behaviours gathered from several low level sensors and modules. These data takes up really small spaces compared to any other data. Thus 32 GB SD card is adequate for the system to work properly.

1.3.1.7 Operations

User-Initiated Operations

- Moving YOLO forward/backward linearly
- Moving YOLO left/right linearly
- Turning YOLO left/right
- Touching YOLO
- Moving YOLO non-linearly
- Stopping YOLO while it moves

Data Processing Support Functions

- Developer can create a new profile

Backup and Recovery Operations

- YOLO waits for response, when it needs to be recharged

1.3.2 System Functions

Function	Summary
Moving YOLO forward/backward linearly	User moves YOLO in a linear route, forward or backward.
Moving YOLO left/right linearly	User moves YOLO in a linear route, left or right.
Turning YOLO left/right	User holds YOLO and change its direction to left or right, while it is moving or fixed in his position.
Touching YOLO	User sends touch inputs via just touching YOLO or holding it.
Moving YOLO non-linearly	User holds YOLO and while moving YOLO, tracks a non-linear route with it.
Stopping YOLO while it moves	User holds YOLO and immobilizes it.
Developer can create a new profile	Developer can set LED light color, animation, time besides movement speed, interval, type, shape and thus YOLO be able to exhibit new behaviours.
Entering the wait for response state	YOLO waits for a response when it needs to be recharged

Table 1.1: System Functions

1.3.3 Stakeholder Characteristics

Users

- The users of YOLO are generally children which are in middle school ages or younger. However, anyone can experience this robot for their free play activities or physical/mental rehabilitation.
- Developers are also using YOLO for their software and system development projects. Developer should be interested in programming with Python and working on Raspbian Stretch Lite OS.

1.3.4 Limitations

a) **Regulatory policies:** The Your Own Living Object (YOLO) is an open-source project licensed under CC-By Attribution 4.0 International open source license and accessible for everyone.[3]

b) **Hardware limitations:** YOLO should be able to process data acquired from its sensors and write them to the memory. Thus, SD card should be able to read and write data. Moreover, Raspberry-pi's Wi-Fi module should run properly to receive created profiles by a developer.

[3] https://github.com/ElsevierSoftwareX/SOFTX˙2019˙242

c) **Interfaces to other applications:** YOLO should be compatible with the API which it establish a connection with developer.

d) **Parallel operation:** No multiple operations in a moment is required. However, YOLO should be operable by multiple users and track the behaviours of those users.

e) **Audit Functions:** YOLO does not have any audit functions.

f) **Control Functions:** Since YOLO is an open source project that can be accessible by community, it does not have any control functions.

g) **Higher-order language requirements:** System is programmed with Python, which is a high-level programming language. The operating system of YOLO is Raspberry Pi OS Lite, which is a Linux distribution. The API used for communication with YOLO should be designed to be compatible with those elements.

h) **Signal handshake protocols:** A Local Area Network (LAN) connection and IP networking protocol are required for creating new profiles for YOLO.

i) **Quality requirements:** Robustness, working stability, and ease of use are required for a qualified usage and experience of YOLO. It should be persistent to physical impacts, not be unstable while working in same conditions and be easy to use for a children, which is the target audience.

j) **Criticality of the application:** YOLO is not a critical system. It is for free play activities, and does not have huge effects in case of failure.

k) **Safety and security considerations:** YOLO is an open source project and the source code of it is accessible for everyone. Thus, YOLO can be hacked over network and forced to exhibit dangerous movements.

l) **Physical/Mental considerations:** LED lights and movements of YOLO guides deaf users. Physically or mentally disabled people should have help while using YOLO.

1.4 Definitions

- YOLO: Your Own Living Robot
- API: Application Programming Interface
- Wi-Fi: Wireless Fidelity
- SD Card: Secure Digital Card
- IP: Internet Protocol
- LAN: Local Area Network

2 References

1. https://www.researchgate.net/publication/340367036_YOLO_-_Your_Own_Living_Object
2. https://www.researchgate.net/publication/340911329_Software_architecture_for_YOLO_a_creativity-stimulating_robot
3. https://github.com/ElsevierSoftwareX/SOFTX_2019_242

3 Specific Requirements

3.1 External Interfaces

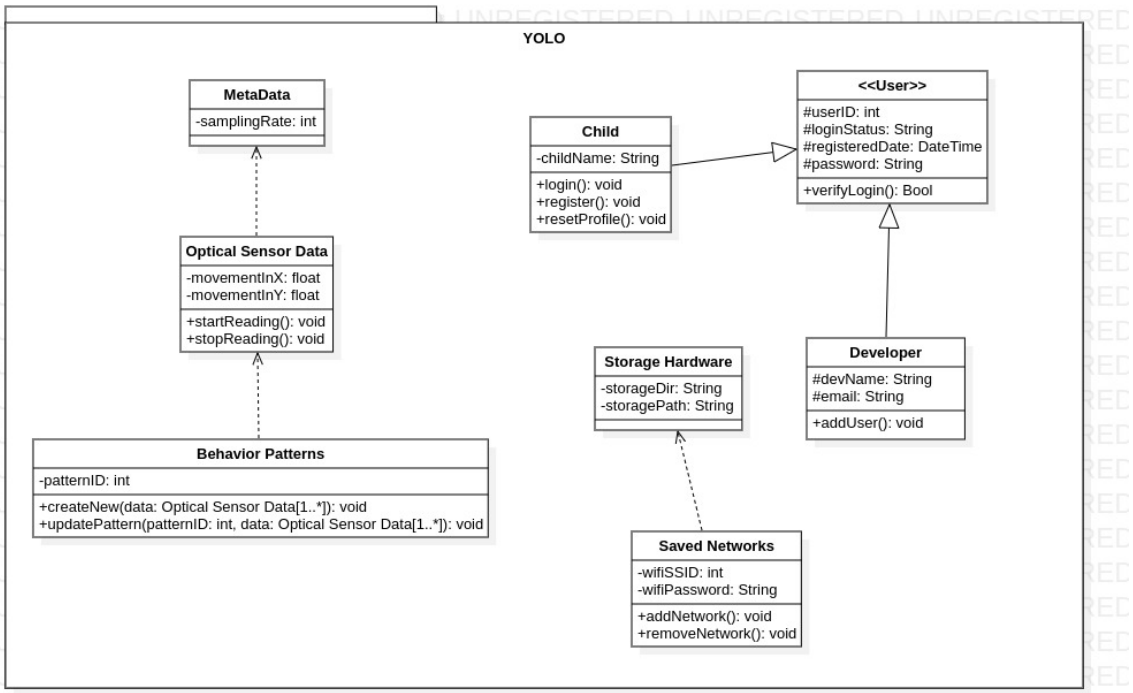

Figure 3.1: External Interfaces Class Diagram

- Child and developer both fall under the category User.
- Network connection details are stored so that the device is able to connect promptly whenever a saved network is in range.
- Behavior patterns are dependent on optical sensor data (shape drawn) which is dependent on the frequency at which images are taken i.e. the sampling rate.

3.2 Functions

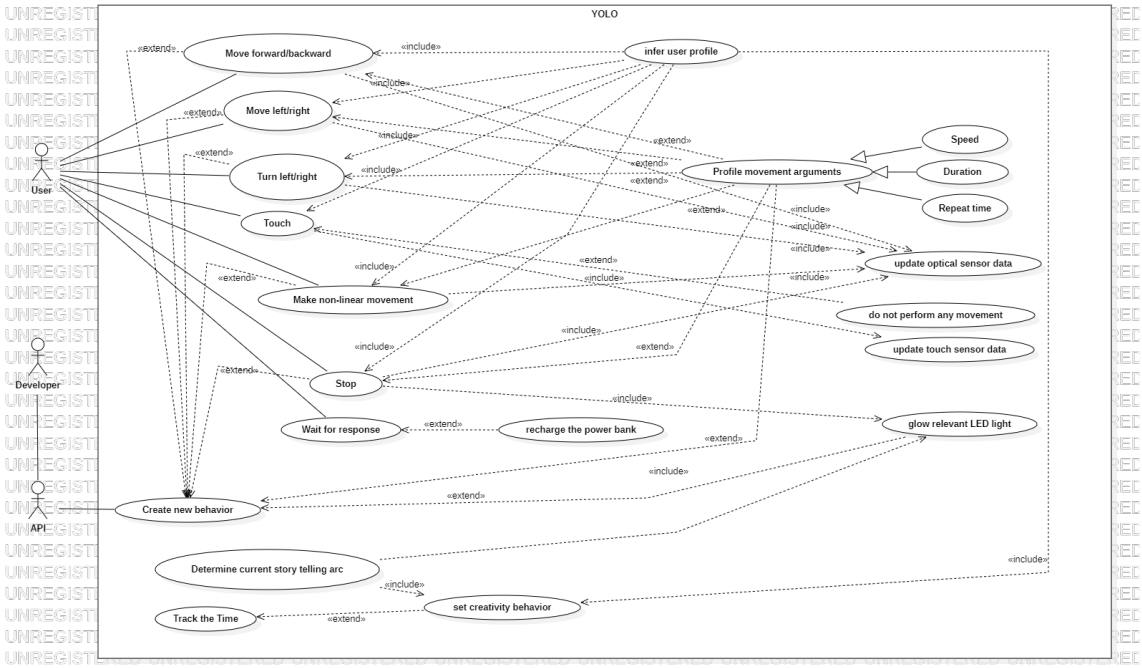

Figure 3.2: Use Case Diagram

Use-Case Name	Move Forward/Backward
Actors	User or Developer
Description	YOLO can be located in another destination by moving linearly forward or backward.
Data	Optical sensor data. For developer, direction, speed, duration and repeat
Preconditions	YOLO must be turned on, run properly and be on the ground with its wheels are in contact with ground and are ready to make a movement.
Stimulus	User physical interaction. Developer sends command over API.
Basic Flow	For User: Step 1 - User makes contact with the YOLO. Step 2 - User pushes or pulls the YOLO in a linear line forward or backward. Step 3 - User releases the YOLO in the target destination. For developer: Step 1 - Developer enters the IP address of YOLO. Step 2 - Developer establishes connection to the YOLO over router. Step 3 - Developer sets the arguments of movement. Step 4 - Developer sends command to the YOLO over API.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	For developer: Step 2 - User enters an invalid IP address or there is no available YOLO devices to establish a connection on the network. Step 3 - Connection is not established. Step 4 - An error occurs and prompt waits for a valid IP address of an available YOLO device.
Post Conditions	The movement path is recorded by the data obtained from optical sensors. Optical sensor data is updated.

Table 3.1: Move Forward/Backward Function

Use-Case Name	Move Left/Right
Actors	User or Developer
Description	YOLO can be located in another destination by moving linearly left or right.
Data	Optical sensor data. For developer, direction, speed, duration and repeat.
Preconditions	YOLO must be turned on, run properly and be on the ground with its wheels are in contact with ground and are ready to make a movement.
Stimulus	User physical interaction. Developer sends command over API.
Basic Flow	For User: Step 1 - User makes contact with the YOLO. Step 2 - User pushes or pulls the YOLO in a linear line left or right. Step 3 - User releases the YOLO in the target destination. For developer: Step 1 - Developer enters the IP address of YOLO. Step 2 - Developer establishes connection to the YOLO over router. Step 3 - Developer sets the arguments of movement. Step 4 - Developer sends command to the YOLO over API.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	For developer: Step 2 - User enters an invalid IP address or there is no available YOLO devices to establish a connection on the network. Step 3 - Connection is not established. Step 4 - An error occurs and prompt waits for a valid IP address of an available YOLO device.
Post Conditions	The movement path is recorded by the data obtained from optical sensors. Optical sensor data is updated.

Table 3.2: Move Left/Right Function

Use-Case Name	Turn Left/Right
Actors	User or Developer
Description	YOLO can be located in the same or another destination with another direction of view to left or right. This move often used in a shaped movements by developer and can be used by user with an arbitrary angle.
Data	Optical sensor data. For developer, with a given shape of movement, the built-in angle of the shape is taken as input data of turn.
Preconditions	YOLO must be turned on, run properly and be on the ground with its wheels are in contact with ground and are ready to make a movement.
Stimulus	User physical interaction. Developer sends command over API.
Basic Flow	For User: Step 1 - User makes contact with the YOLO. Step 2 - User turns the YOLO left or right. Step 3 - User releases the YOLO in the target destination. For developer: Step 1 - Developer enters the IP address of YOLO. Step 2 - Developer establishes connection to the YOLO over router. Step 3 - Developer sets the arguments of movement. Step 4 - Developer sends command to the YOLO over API.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	For developer: Step 2 - User enters an invalid IP address or there is no available YOLO devices to establish a connection on the network. Step 3 - Connection is not established. Step 4 - An error occurs and prompt waits for a valid IP address of an available YOLO device.
Post Conditions	The movement path is recorded by the data obtained from optical sensors. Optical sensor data is updated.

Table 3.3: Turn Left/Right Function

Use-Case Name	Touch
Actors	User
Description	YOLO enters to the waiting condition to get physical inputs from user.
Data	Touch sensor signal.
Preconditions	YOLO must be turned on and run properly.
Stimulus	User physical interaction
Basic Flow	For User: Step 1 - User touches the YOLO.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	Step 2 - YOLO's touch sensor may be unable or damaged and can not receive the physical touch input from user. Step 3 - YOLO does not respond to the touch.
Post Conditions	YOLO is ready to move and waiting for an interaction. Touch sensor data is updated.

Table 3.4: Touch Function

Use-Case Name	Make Non-Linear Movement
Actors	User or Developer
Description	YOLO can be located in another destination by moving non-linearly.
Data	Optical sensor data. For developer, shape of movement, direction, speed, duration and repeat
Preconditions	YOLO must be turned on, run properly and be on the ground with its wheels are in contact with ground and are ready to make a movement.
Stimulus	User physical interaction. A push or a pull to the YOLO Developer sends command over API.
Basic Flow	For User: Step 1 - User makes contact with the YOLO. Step 2 - User pushes or pulls the YOLO in a non-linear way. Step 3 - User releases the YOLO in the target destination. For developer: Step 1 - Developer enters the IP address of YOLO. Step 2 - Developer establishes connection to the YOLO over router. Step 3 - Developer sets the arguments of movement with a shape argument included. Step 4 - Developer sends command to the YOLO over API.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	For developer: Step 2 - User enters an invalid IP address or there is no available YOLO devices to establish a connection on the network. Step 3 - Connection is not established. Step 4 - An error occurs and prompt waits for a valid IP address of an available YOLO device.
Post Conditions	The movement path is recorded by the data obtained from optical sensors. Optical sensor data is updated.

Table 3.5: Make Non-Linear Movement Function

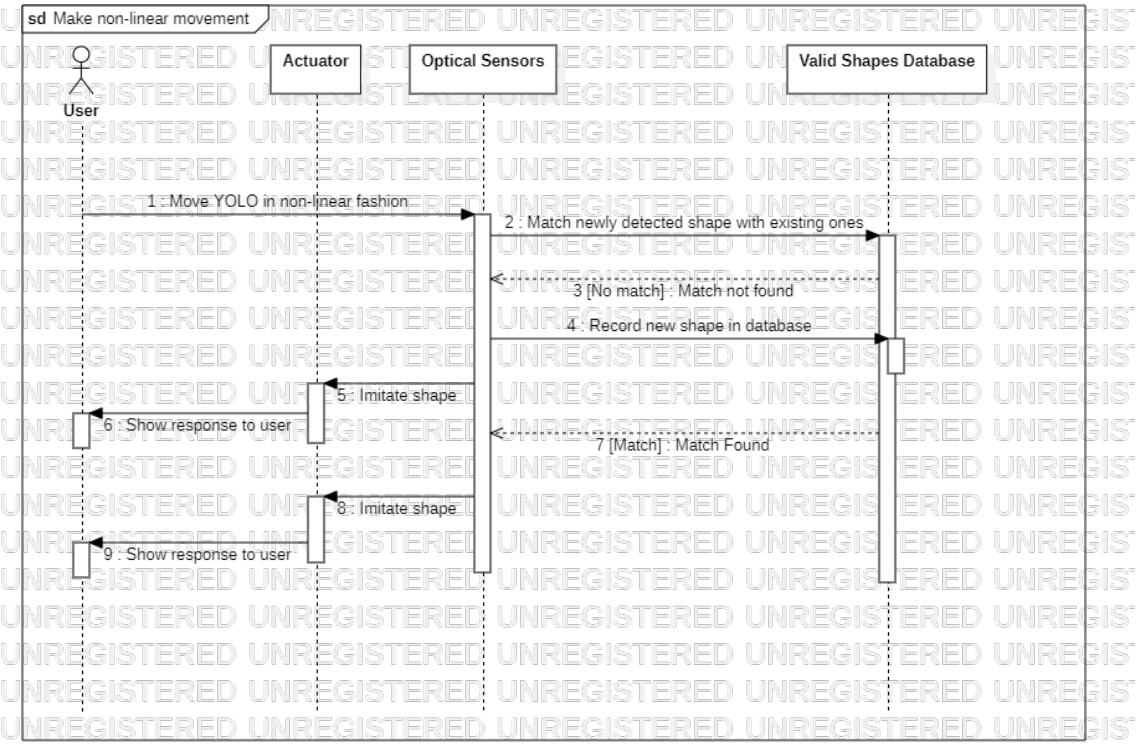

Figure 3.3: Sequence Diagram for Make Non-linear Movement Use Case

Use-Case Name	Stop
Actors	User or Developer
Description	YOLO's movement is interfered and it is in a stable status.
Data	Optical sensor data. For developer, shape of movement, direction, speed, duration and repeat
Preconditions	YOLO must be turned on, run properly and be on the ground with its wheels are in contact with ground and are ready to make a movement.
Stimulus	User physical interaction. A push or a pull to the YOLO Developer sends command over API.
Basic Flow	For User: Step 1 - User makes contact with the YOLO. Step 2 - User pushes or pulls the YOLO to immobilize it. Step 3 - User releases the YOLO in the target destination. For developer: Step 1 - Developer enters the IP address of YOLO. Step 2 - Developer establishes connection to the YOLO over router. Step 3 - Developer sets the arguments of movement with a speed argument equals to zero meter per seconds. Step 4 - Developer sends command to the YOLO over API.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	For developer: Step 2 - User enters an invalid IP address or there is no available YOLO devices to establish a connection on the network. Step 3 - Connection is not established. Step 4 - An error occurs and prompt waits for a valid IP address of an available YOLO device.
Post Conditions	The movement path until the immobilization is recorded by the data obtained from optical sensors. Optical sensor data is updated. Corresponding LED light is lit.

Table 3.6: Stop Function

Use-Case Name	Wait for Response
Actors	User
Description	Upon interacting with the robot, for example by making it move, the user expects some response from it too. However, in case the robot's battery has depleted, it would not respond and would need to be recharged.
Data	-
Preconditions	-
Stimulus	User physical interaction. A push or a pull to the YOLO.
Basic Flow	For User: Step 1 - User makes contact with the YOLO. Step 2 - User pushes or pulls the YOLO in an arbitrary way. Step 3 - User releases the YOLO in the target destination. Step 4 - YOLO gives response if its battery is not run out.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	Step 5 - YOLO does not give any response. Its battery is run out and must be handled under adult supervision.
Post Conditions	Glowing of LED lights or movement response.

Table 3.7: Wait for Response Function

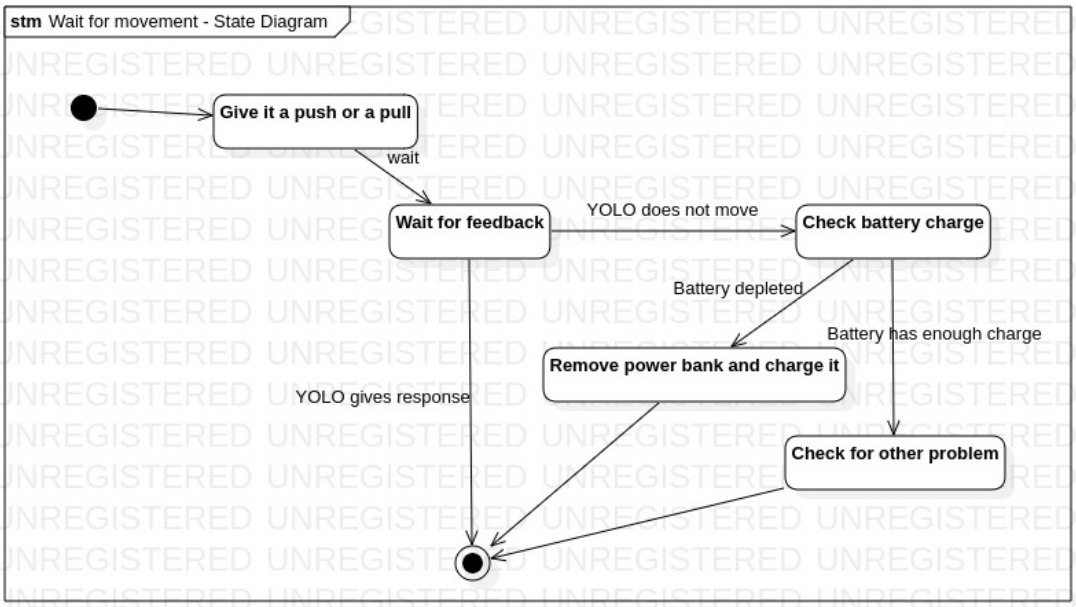

Figure 3.4: State Diagram for Wait For Response Use Case

Use-Case Name	Create New Behavior
Actors	Developer
Description	The behavior type of the YOLO can be modified by a developer to create more personalized experience while using the robot.
Data	Name of the behavior, new waiting time, direction, shape, duration, speed and repeat count of the movement and LED light RGB, brightness, animation, duration and repeat count variables.
Preconditions	-
Stimulus	Developer command received over API.
Basic Flow	For developer: Step 1 - Developer enters the IP address of YOLO. Step 2 - Developer establishes connection to the YOLO over router. Step 3 - Developer sets the arguments of new behavior type. Step 4 - Developer sends command to the YOLO over API.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	For developer: Step 2 - User enters an invalid IP address or there is no available YOLO devices to establish a connection on the network. Step 3 - Connection is not established. Step 4 - An error occurs and prompt waits for a valid IP address of an available YOLO device.
Post Conditions	Ready to exhibit different behaviors than before. Corresponding LED light is lit.

Table 3.8: Create New Behavior Function

Use-Case Name	Determine Current Story Telling Arc
Actors	-
Description	Using machine learning (ML), the software will try to infer which user, among multiple saved users, is currently using the robot. It will also use ML to set the creativity behavior to contrasting or mirroring. This is provided by K-nearest neighbour (KNN) algorithm.
Data	Usage history obtained from previous movements of a user.
Preconditions	Being used by at least one user before.
Stimulus	By moving YOLO, drawing patterns and then waiting for the it to respond.
Basic Flow	Step 1 - YOLO gathers data from previous movements. Step 2 - YOLO processes the data with KNN algorithm. Step 3 - YOLO makes inferences about user. Step 4 - YOLO update its creativity behavior according to the user.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	Step 2 - There is not enough data for YOLO to make precise inferences for a user.
Post Conditions	YOLO is ready to act with different creativity behaviors for each user and determines current story telling arc accordingly. Corresponding LED light is lit.

Table 3.9: Determine Current Story Telling Arc Function

Use-Case Name	Track the Time
Actors	-
Description	The software shall track the time so it may monitor the current creativity stage. If enough time has elapsed, it would switch to the next creativity stage.
Data	Time
Preconditions	YOLO must be turned on, run properly and having a sequence of different movements with or without a stop condition.
Stimulus	-
Basic Flow	Step 1 - YOLO is making arbitrary movements. Step 2 - Time measurement starts. Step 3 - YOLO checks whether enough time has elapsed. Step 4 - YOLO switches to the next creativity stage acoordingly.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	Step 2 - YOLO is continually stationary.
Post Conditions	A different creativity stage would result in a different behavior of the YOLO in response to the same movements by the user.

Table 3.10: Track the Time Function

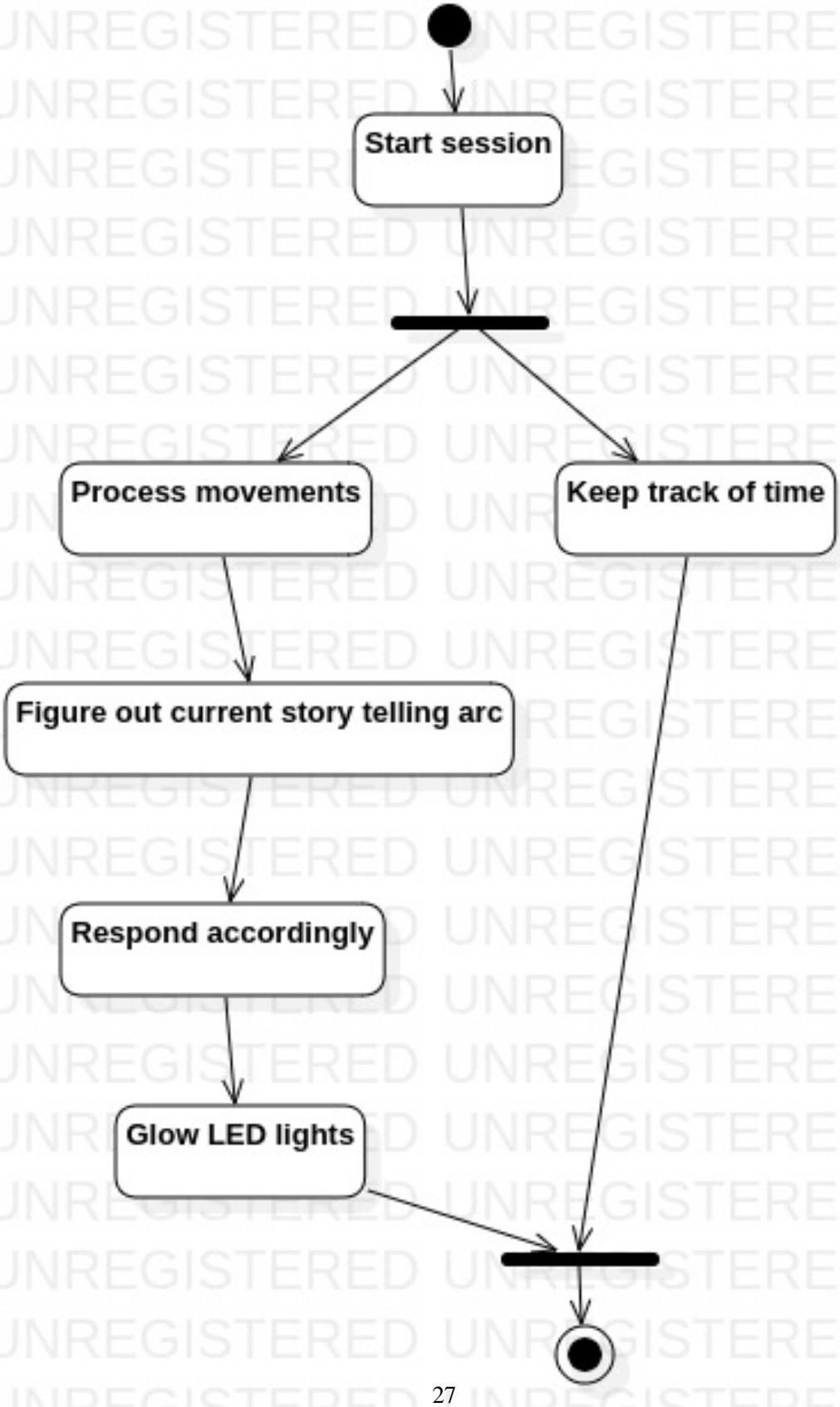

Figure 3.5: Activity Diagram for Create New Behavior Use Case

3.3 Usability Requirements

- Upon start-up with the help of a supervisor, the device should produce a stimulus in the form of glowing LED lights for a second to indicate that it is ready for input.
- User should get response according to the time s/he has already spent playing and according to the current story-telling arc.
- Developers should be able to change the behavior and add more functionality to the system.
- Play history must be saved in the storage so that the data can be used to make better decisions about child's playful habits.

3.4 Performance Requirements

- Battery level should not go below 10 percent.
- User should receive feedback withing 5 seconds after stopping interaction with the device.
- Only one child should play with one device at a time.
- Optical sensors and LED lights should be in working condition.
- If size of user specific data exceeds storage size, start storing it on the connected computer and display relevant message when user logins i.e upgrade storage.

3.5 Logical Database Requirements

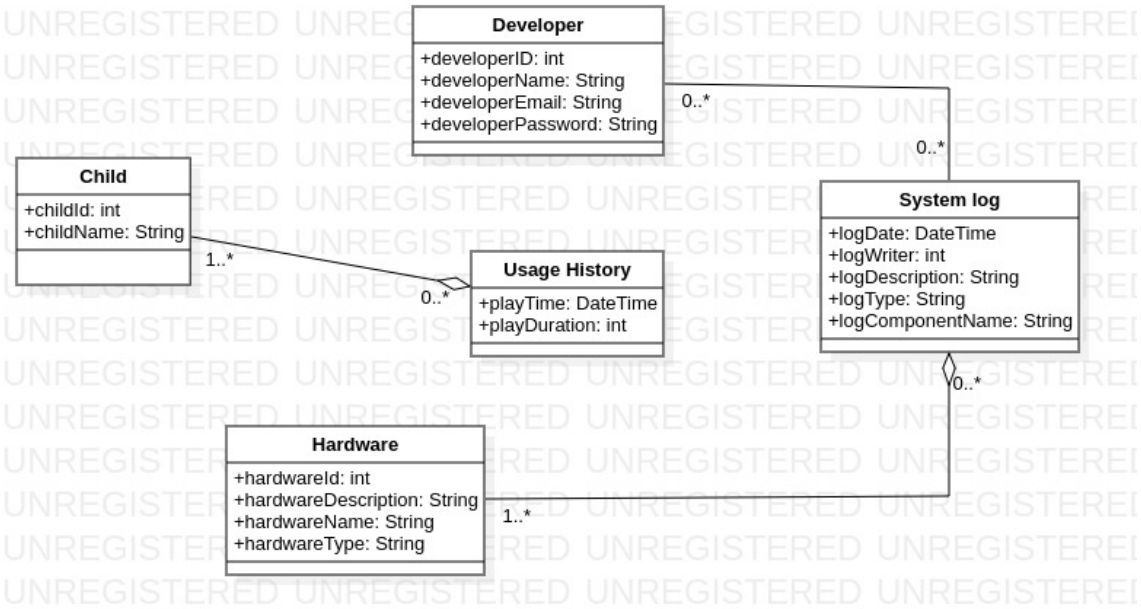

Figure 3.6: Logical Database Requirements Class Diagram

- Database should be loaded when device is started.
- Every child is identified by a unique id so that there is no ambiguity in case of two or more children with the same name.
- Child ID is displayed on computer screen when a new child is registered.
- Usage history is tracked for every child as to the last day and time the child used the device and how long it was used for.
- Developers need to have their password ready in order to log-in and make changes.
- Database must be constantly updated while device is in use.
- In case of some hardware failure, system logs should be updated with specific information so that diagnosing the problem is easy.
- System log data should be read every time the device starts and appropriate messages should be sent to the user via the computer's screen.

3.6 Design Constraints

- Database data should be password-protected so that private information of users and developers such as their name, email e.t.c. cannot be accessed by any unauthorized person.
- Source code should be written in python.
- Open source hardware design and open source software result in a cost-effective product.

3.7 System Attributes

a) **Reliability**

- Data shall always be backed up to the cloud at the end of a session of play.

b) **Availability**

- Failure of the system's components should be less than 10 minutes in a month.

c) **Security**

- Source code cannot be modified unless an authorized developer logs in with his/her credentials.

d) **Maintainability**

- Software can be updated easily from the terminal.
- Hardware for example the battery, sensors e.t.c. can be upgraded provided that the new components fit in properly.

e) **Portability**

- System can be used wherever there is an internet connection.

3.8 Supporting Information

YOLO is an open-source project which aims to uncover new horizons of the branch of Computer Science called human-computer-interaction. Information about materials used in the project including source code is available on the project's GitHub[4] repository. In addition to this, people with technical knowledge can contribute to the project.

[4] https://github.com/ElsevierSoftwareX/SOFTX˙2019˙242

4 Suggestions to Improve the Existing System

- Infrared sensor, sound sensors and buzzers shall be added for blind people and their community to interact easily with YOLO. Their specific situation is requiring these sensor and buzzers with an appropriate software reorganizations, such as programming the device to sound periodically according to the distance to the user obtained from infrared sensors and processed by an algorithm. As YOLO gets closer to the user, the buzzer makes noisies/sounds more frequently.
  With this improvement on the system, also people who lost their sight afterwards can be rehabilitated mentally and trained in order to improve their sensitivity to sounds.
  Those improvements will increase the accessibility of YOLO by everyone, especially disabled people.

- An RFID reader can be integrated into YOLO. Thus, YOLO will be user-specific device and can not be run by anyone else, or can not be forced to make specificed movements and acts with a minimal software reorganization. These constraints help user to have a sovereignty over their device.