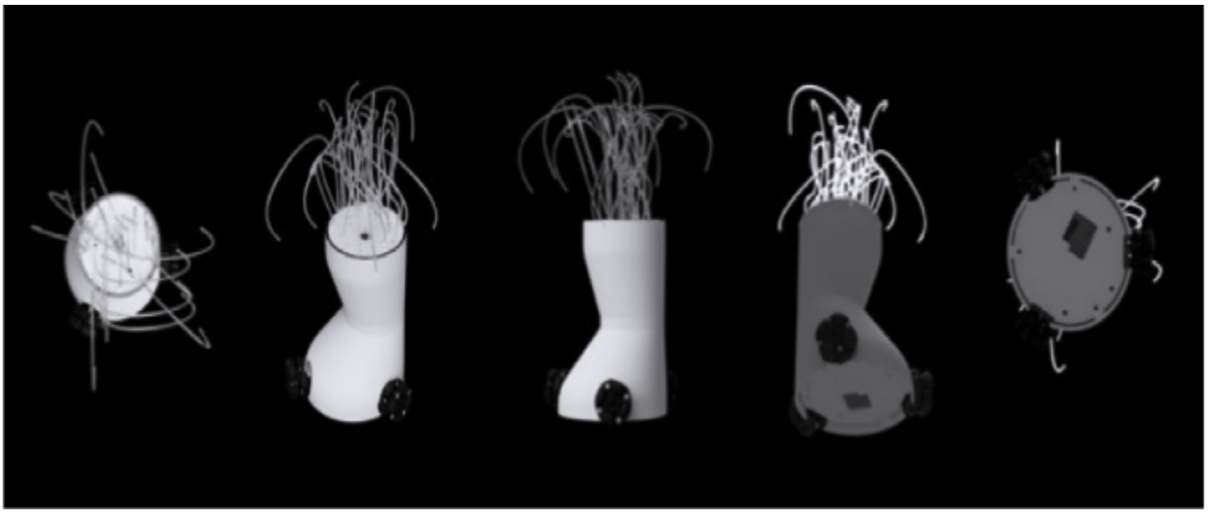
YOLO

(Your Own Living Object)



Software Requirements Specification

Version 1.1

Berkay Bartug Cetin 2309839

Anil Berdan Ceylan 2304277

Table of Contents

List of Figures

List of Tables

Revision History

Version	Date	Explanation
1.0	07.04.2022	Initial use cases are determined. Context diagram is drawn. Project purpose, scope and general information is added.
1.0	15.04.2022	Rest of the document is prepared. Some of the use cases and context diagram entities are rewritten.

Table 1: Revision History of Software Requirements Specification Document

1 Introduction

This document is the Software Specification Requirement (SRS) of a social-smart robot which is YOLO (Your Own Living Object) developed by four researchers.

1.1 Purpose

The purpose of this project is to create -a new generation of technological toy- a smart robot which can interact with children and help them improve their creativity. The main idea is to increase the creative thought process of children by interacting with them by using alternative but effective interactive modalities.

1.2 Scope

YOLO (Your Own Living Object) is a social robot designed to foster creativity in youngsters through storytelling activities. It appears in children's literature as a character. The YOLO artificial intelligence software tests a range of Creativity Behaviors to see which ones are the most effective in stimulating creativity. YOLO can choose between two different types of creative thinking: convergent and divergent thinking. These tactics were developed based on psychological theories of creativity development as well as research from experts in the field of children's creativity.

In addition to fostering creativity, this program allows for the creation of Social Behaviors, which allow the robot to behave like a real human. The three primary social behavior qualities that emerged are exuberant, aloof, and harmonious.

These features are meant to facilitate immersive gameplay and character development. During co-design research, the three social behaviors were developed with the cooperation of children and were founded on psychological theories of personality.

In general, this paper examines the design, development, and deployment of social robots that can assist individuals in developing natural human characteristics such as creativity.

1.3 Product Overview

This section of the document will provide detailed information about the system including all components.

1.3.1 Product Perspective

YOLO (Your Own Living Object) is neither a part of a large system nor a simple toy. YOLO is a social robot designed and developed to stimulate creativity in children through storytelling activities. Children use it as a character in their stories and use it as a helper to their creative thought process.

The software for the YOLO hardware should be exclusively installed on YOLO hardware. All customers shall follow the installation guide in order to install the software of their own living object's Raspberry-Pi. YOLO uses different kinds of sensors and camera management systems to get the 0data that is used for learning the environment and reacting to the outer world.
(YOLO Software website is https://github.com/patricialvesoliveira/YOLO-Software and the installation guide can be found in https://github.com/patricialvesoliveira/YOLO-Software/wiki)

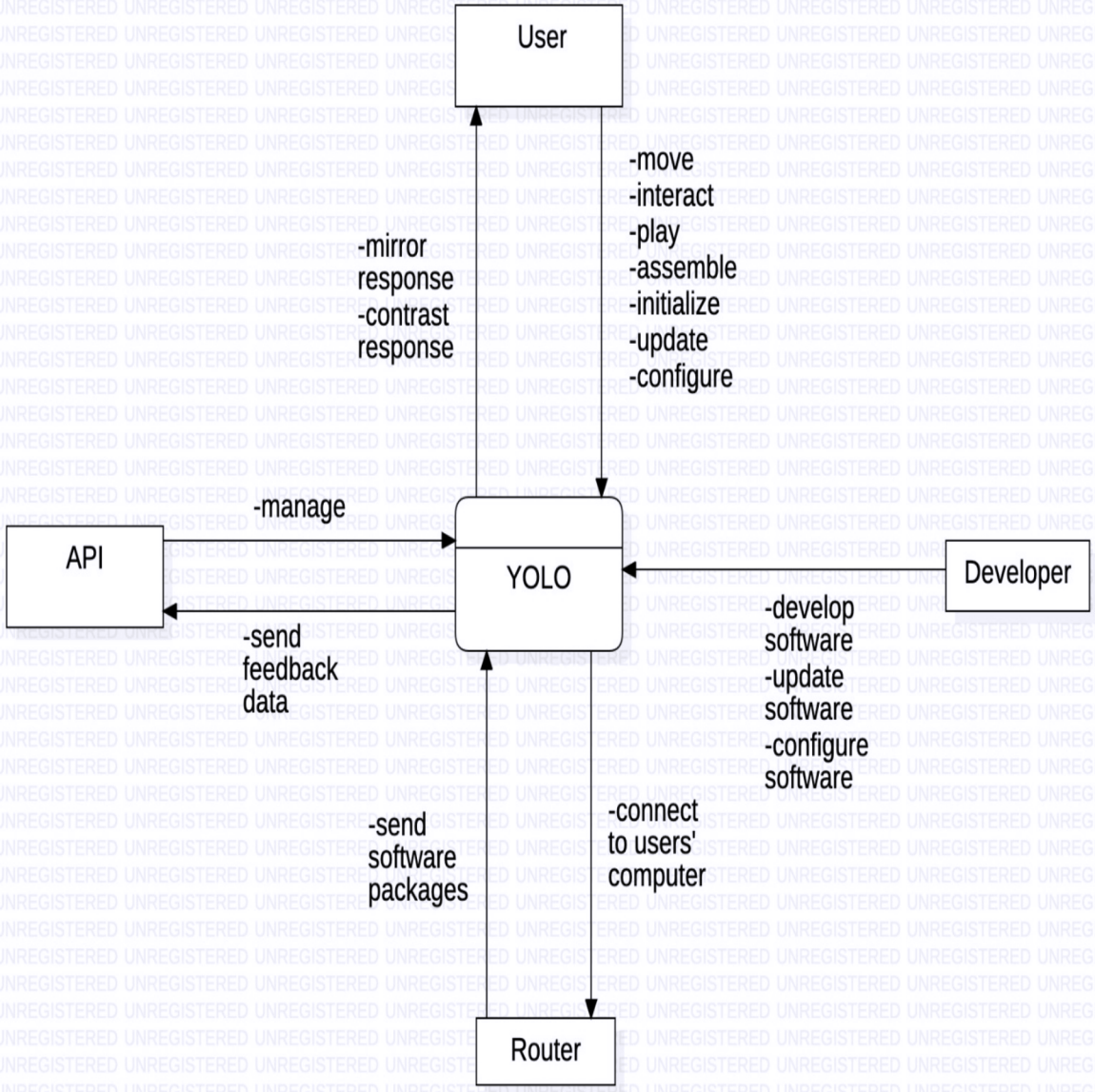


Figure 1: Context Diagram for YOLO

1.3.2 Product Functions

Functionalities (use cases) of Your Own Living Object (YOLO) are summarized as what they do below. More advanced and detailed versions can be found in **Functions section (3.2)** with their complete description tables.

Functions	**Summary**
Assemble YOLO	YOLO is not a mass production smart-robot, it can be created by producing necessary hardware parts and assembling them according to the manual.
Initialize YOLO with default software	In order to make YOLO responsive and operate it, a specific software should be downloaded and installed to the Raspberry-Pi System with Python code.
Initialize YOLO with custom software	In order to operate YOLO in a custom way, a custom software should be downloaded and installed to the Raspberry-Pi System with Python code.
Configure YOLO	YOLO's software is open source and it can be changed according to the child and parent's wishes for new behaviors.
Update YOLO software	YOLO software is open source and parents can use different codes to install YOLO's Raspberry-Pi System, which are created by other developers.
Interact with YOLO via touching	YOLO gives different output behaviors to different touching patterns, hence children can interact with it in various ways.

Interact with YOLO by moving it	YOLO gives different output behaviors to different moving patterns, hence children can interact with it in various ways.
Play with YOLO	After a child gets used to YOLO and the way that it interacts, the child starts playing with it and accepts it as a friend.
Use YOLO's API to study child-robot interaction	Researchers are able to examine how different APIs cause various kinds of output behaviors and study child-robot interactions by using this data.
Develop Software for YOLO	Since YOLO software is open source, it is programmable and it can be used to see various output behaviors.

Table 2: System Functions for YOLO

1.3.3 User Characteristics

Your Own Living Object (YOLO) is an intelligent robot developed to increase children's creative thought processes by interacting with them. Hence the target users of YOLO are just children but as children might not be able to create their own YOLO from the given instruction papers, parents of the children might need to use it. Moreover, as YOLO is a robot that can be used to study child-robot interaction by researchers and because of its open-source software and programmable hardware, teachers can use it in an educational aspect. Therefore, children should know how to turn on and turn off YOLO and learn the basic behavior patterns that it uses to interact with them for playing with it according to their wishes. Parents should have at least enough knowledge to assemble and initialize YOLO for their children while researchers and teachers who want to use YOLO as a part of a study or an educational tool should have enough knowledge to make changes in YOLO software to operate YOLO and gather information for their needs.

1.3.4 Limitations

- **Regulatory policies:** Your Own Living Object is an interactive smart robot and if it is used as a research tool for studying child-robot interaction, data that is gathered from that research must be kept private.

- **Hardware limitations:** The hardware used in YOLO to store software is a Raspberry Pi system that has space restrictions for the portable usage which makes the hardware options limited.

- **Interface to other applications:** For the control and configuration of YOLO, a Raspberry Pi intercafe is needed.

- **Parallel operation:** YOLO must be able to process different tasks at the same time such as detecting motion direction and detecting physical touches.

- **Audit functions:** There is no audit function in the product.

- **Control functions:** The control of the YOLO is limited with the control system which includes actuators and sensors.

- **Higher-order language requirements:** YOLO includes a Raspberry Pi to run machine learning tasks and operate the hardware according to sensor inputs, so knowing the basics of Python might come handy in case of a software problem.

- **Signal handshake protocols:** There is no such limitation.

- **Quality requirements:** There is no such limitation.

- **Criticality of the application:** There is no such limitation.

- **Safety and security considerations:** Due to the material used and might be used for assembling YOLO, high temperature objects are not advised to be inspected.

- **Physical/mental considerations:** There is no such limitation.

- **Limitations that are sourced from other systems:** There is no such limitation.

1.4 Definitions

Term	Definition
API	Application Programming Interface
Raspberry Pi	Series of small single board computers. It makes YOLO programmable and responsive.

Table 3: Definitions

2 References

This document is written with respect to the specifications of the document below:

29148-2018 - ISO/IEC/IEEE International Standard - Systems and software engineering– Life cycle processes –Requirements engineering.

Other sources:

- Alves-Oliveira, P., Gomes, S., Chandak, A., Arriaga, P., Hoffman, G., & Paiva, A. (2020b). Software architecture for YOLO, a creativity-stimulating robot. SoftwareX, 11, 100461. https://doi.org/10.1016/j.softx.2020.100461

- Alves-Oliveira, P., Arriaga, P., Paiva, A., & Hoffman, G. (2019). Guide to build YOLO, a creativity-stimulating robot for children. *HardwareX*, *6*, e00074. https://doi.org/10.1016/j.ohx.2019.e00074

- Patrícia Alves-Oliveira, Patrícia Arriaga, Ana Paiva, and Guy Hofman. 2021. Children as Robot Designers. In Proceedings of the 2021 ACM/IEEE

International Conference on Human-Robot Interaction (HRI ’21), March 8–11, 2021, Boulder, CO, USA. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3434073.3444650

3 Specific Requirements

This section of the document will provide detailed information about the specific requirements of Your Own Living Object (YOLO).

3.1 External Interfaces

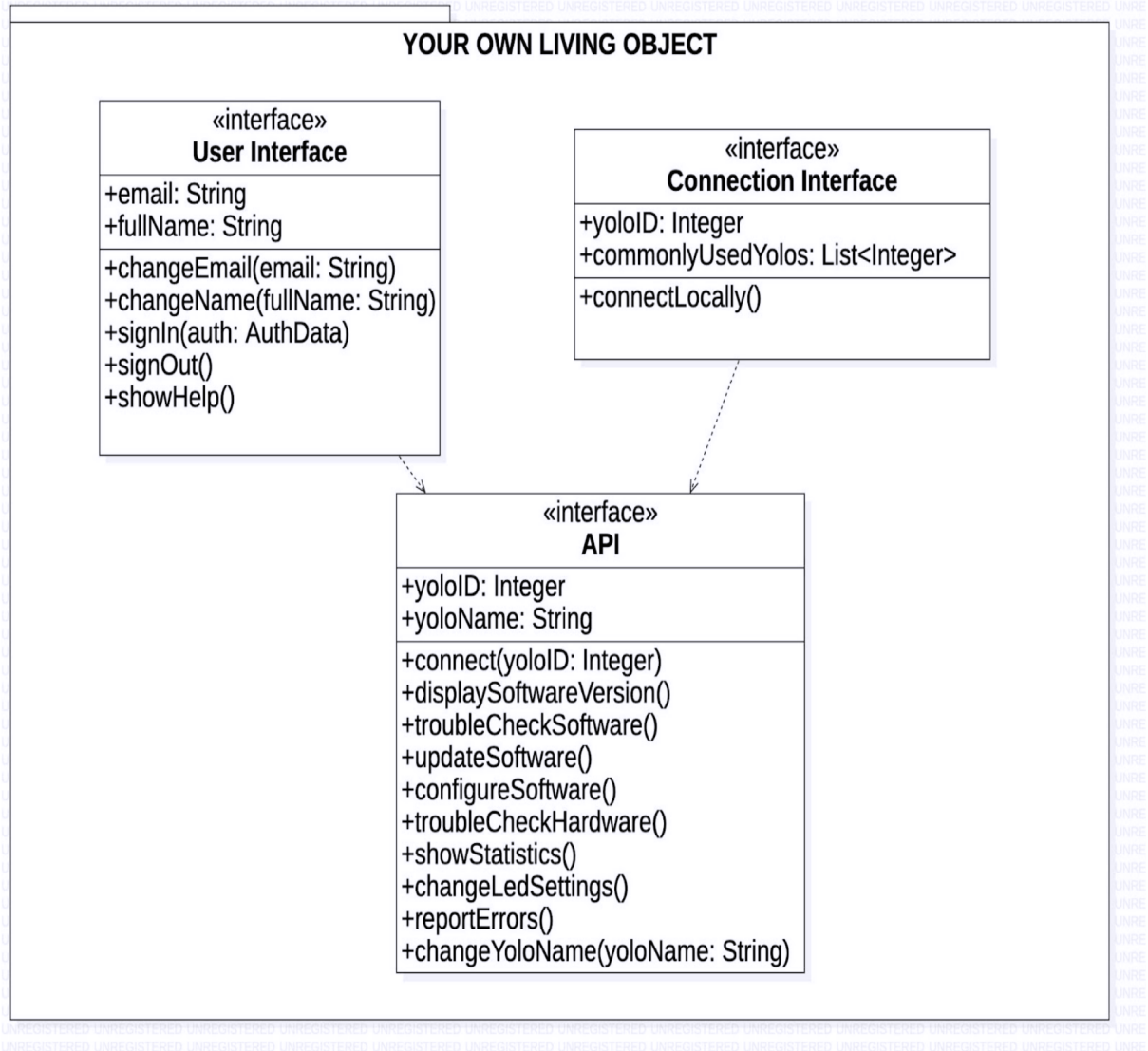


Figure 2: External Interface Diagram for YOLO

3.2 Functions

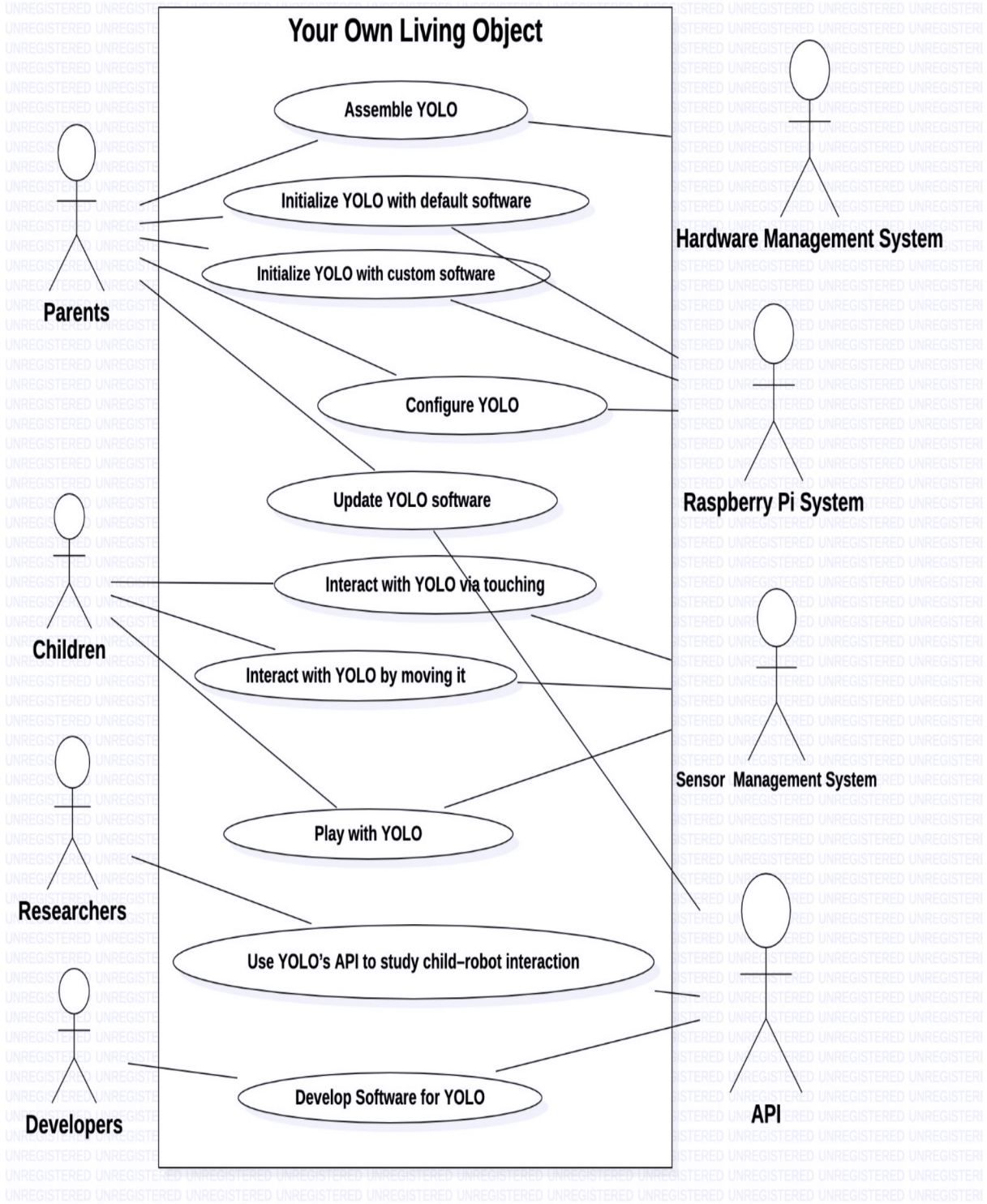


Figure 3: Use Case Diagram for YOLO

Use case name	Assemble YOLO
Actors	Parent, Hardware Management System
Description	To initialize YOLO, first it must be assembled according to given steps
Data	The manual
Preconditions	Need to have YOLO hardware parts
Stimulus	Parent assembles YOLO parts
Basic Flow	Step 1: Heat M2 brass inserts on the boards layer and wheels layer Step 2: Heat M3 brass inserts on the tabs at the bottom of the shell Step 3: Perform wire connections Step 4: Screw the raspberry-pi, touch sensor, and the power distribution board to the boards layer using M2 pan head screws Step 5: Slide the 9V battery and the power bank into the battery layer Step 6: Screw the glowing fibers layer to the jewel layer using washers and M2 pan head screws Step 7: Screw the omni wheels to the motor using socket head screws Step 8: Check hardware system for being sure about compatibility of each system component by testing them individually
Alternative Flow	-
Exception Flow	**-**
Postconditions	YOLO robot is ready to initialize

Table 4: Assemble YOLO

Use case name	Initialize YOLO with default software
Actors	Parent, Raspberry-Pi System
Description	To operate YOLO a specific software should be downloaded and installed to the Raspberry-Pi System with Python code.
Data	The software code written in Python
Preconditions	YOLO must be assembled.
Stimulus	Parent initializes YOLO
Basic Flow	Step 1: Parent initializes YOLO's software programme. Step 2: Parent initializes YOLO's software programme. Step 3: Parent connects Raspberry-Pi System and the software programme via wi-fi
Alternative Flow	Step 1: Parent creates a new Python code file according to specifications
Exception Flow	-
Postconditions	After following these steps, YOLO is activated.

Table 5: Initialize YOLO with default software

Use case name	Initialize YOLO with custom software
Actors	Parent, Raspberry-Pi System
Description	To operate YOLO configured software should be downloaded to Raspberry-Pi System wıth Python code.
Data	The software code written in Python
Preconditions	YOLO must be assembled.
Stimulus	Parent initializes YOLO with configured software
Basic Flow	Step 1: Parent creates a new Python code file according to specifications Step 2: Parent initializes YOLO's software programme. Step 3: Parent connects Raspberry-Pi System and the software programme via wi-fi
Alternative Flow	Step 1: Parent initializes YOLO's software programme.
Exception Flow	-
Postconditions	After following these steps, YOLO is activated according to the newly created Python file.

Table 6: Initialize YOLO with custom software

Use case name	Configure YOLO software
Actors	Parent, Raspberry-Pi System
Description	Since YOLO software is open source, it can be changed according to the child and parent's wishes for new behaviors.
Data	The software code written in Python by the parent
Preconditions	YOLO must be assembled and initialized
Stimulus	Parent changes the software inside the YOLO`s Raspberry-Pi System
Basic Flow	Step 1: Parent creates a new Python source code file according to specifications Step 2: Parent changes the source code in the Raspberry-Pi System Step 3: Parent initializes YOLO's software programme. Step 4: Parent connects Raspberry-Pi System and the software programme via wi-fi
Alternative Flow	Step 1: Parent choose another source file that created by another user before
Exception Flow	**Step 2: Code invalid**
Postconditions	After following these steps, YOLO is activated and gives different output behaviors.

Table 7: Configure YOLO software

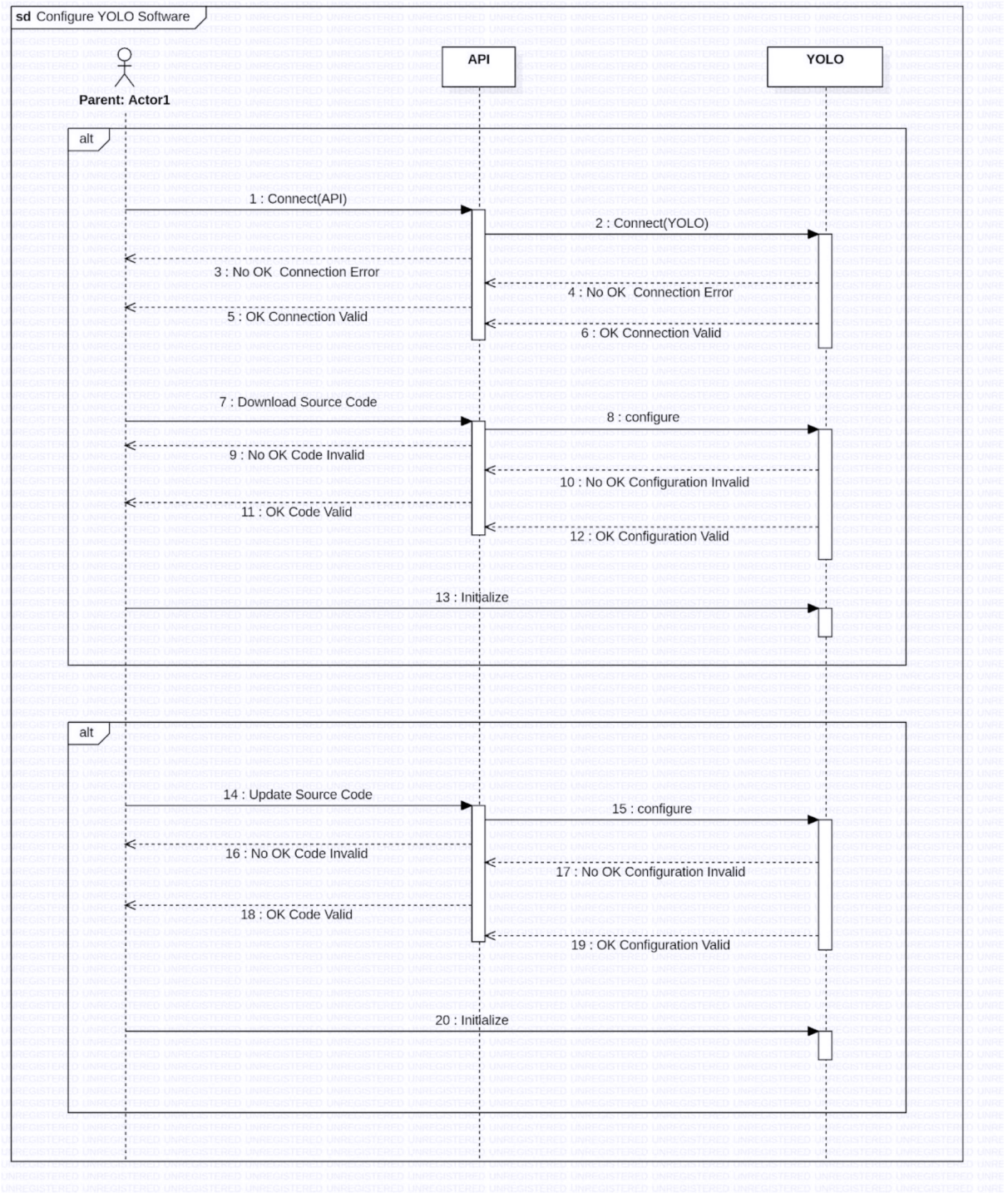


Figure 4: Sequence diagram of Configure YOLO software

Use case name	Update YOLO software
Actors	Parent, API
Description	Since YOLO software is open source, parent can use different codes which are created by another people
Data	The software code written in Python
Preconditions	YOLO must be assembled and initialized
Stimulus	Parent changes the software inside the YOLO`s Raspberry-Pi System
Basic Flow	Step 1: Parent choses a new Python source code file from Github Step 2: Parent changes the source code in the Raspberry-Pi System Step 3: Parent initializes YOLO's software programme. Step 4: Parent connects Raspberry-Pi System and the software programme via wi-fi
Alternative Flow	-
Exception Flow	**-**
Postconditions	After following these steps, YOLO is activated and gives different output behaviors.

Table 8: Update YOLO software

Use case name	Interact with YOLO via touching
Actors	Child, Sensor Management System
Description	YOLO gives different output behaviors to different touching patterns
Data	Touching
Preconditions	YOLO must be activated
Stimulus	Child touches to YOLO or doesn`t touch and see different results according to touching pattern
Basic Flow	Step 1: Child plays with YOLO Step 2: Touching sensors sense the playtime has begun Step 3: Machine learning system analyzes the pattern Step 4: Machine learning system creates a storytelling arc Step 5: Child stops touching to the YOLO Step 6: YOLO responds according the created storytelling arc`s rising action phase Step 7: YOLO stimulates convergent thinking using mirror creativity technique
Alternative Flow	Step 6: YOLO responds according the created storytelling arc`s climax phase Step 7: YOLO stimulates divergent thinking using contrast creativity technique
Exception Flow	-
Postconditions	YOLO gives output to boost creativity in both divergent and convergent thinking

Table 9: Interact with YOLO via touching

Use case name	Interact with YOLO via by moving it
Actors	Child, Sensor Management System
Description	YOLO gives different output behaviors to different moving patterns
Data	Child`s moves with YOLO
Preconditions	YOLO must be activated
Stimulus	Child plays with YOLO
Basic Flow	Step 1: Child plays with YOLO by moving it on the floor Step 2: Motion sensors senses patterns according to child`s moves Step 3: Machine learning system analyzes the pattern Step 4: Machine learning system creates a storytelling arc Step 5: Child stops playing with YOLO Step 6: YOLO responds according the created storytelling arc`s rising action phase Step 7: YOLO stimulates convergent thinking using mirror creativity technique
Alternative Flow	Step 6: YOLO responds according the created storytelling arc`s climax phase Step 7: YOLO stimulates divergent thinking using contrast creativity technique
Exception Flow	-
Postconditions	YOLO gives output to boost creativity in both divergent and convergent thinking

Table 10: Interact with YOLO via by moving it

Use case name	Play with YOLO
Actors	Child, Sensor Management System
Description	After adapting to each other, YOLO becomes a friend to child
Data	Playing patterns of child
Preconditions	YOLO must be assembled and initialized
Stimulus	Child adapts to YOLO
Basic Flow	Step 1: Child plays with YOLO Step 2: Sensors senses patterns according to child`s moves Step 3: Machine learning system analyzes the patterns Step 4: Machine learning system creates a storytelling arc Step 6: YOLO responds according the created storytelling arc`s rising action phase Step 7: YOLO stimulates convergent thinking using mirror creativity technique Step 8 : Child and YOLO adapt to each other Step 9: With the analyzed patterns YOLO helps child to boost creativity
Alternative Flow	Step 6: YOLO responds according the created storytelling arc`s climax phase Step 7: YOLO stimulates divergent thinking using contrast creativity technique
Exception Flow	-
Postconditions	After following these steps, YOLO is activated and gives different output behaviors.

Table 11: Play with YOLO

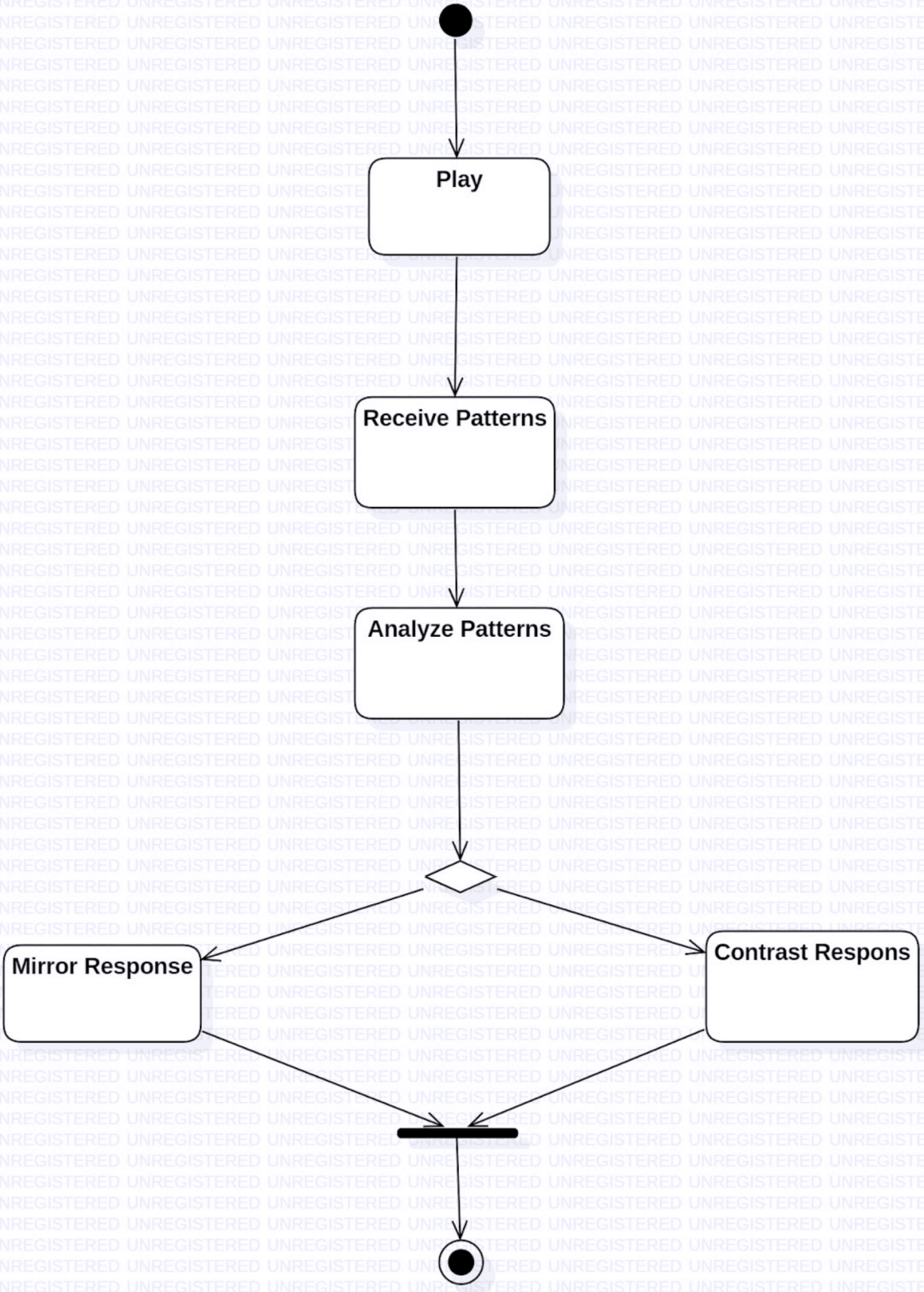


Figure 5: Activity Diagram for Play with YOLO

Use case name	Use YOLO`s API to study child-robot interaction
Actors	Researcher, API
Description	Researchers can examine how different APIs cause various kinds of output behaviors and study child-robot interactions by using this data.
Data	YOLO`s outputs and child's behaviors
Preconditions	Need to have different APIs
Stimulus	Researcher tries different APIs to see distinct results
Basic Flow	Step 1: Researcher chooses different types of YOLO software Step 2: Researcher initializes YOLO's software programme. Step 4: Researcher connects Raspberry-Pi System and the software programme via wi-fi Step 5: Child plays with YOLO Step 6: Sensors senses patterns according to child`s moves Step 7: Machine learning system analyzes the pattern Step 8: Machine learning system creates a storytelling arc Step 9: Child stops playing with YOLO Step 10: YOLO responds according the created storytelling arc`s rising action phase Step 11: YOLO stimulates convergent thinking using mirror creativity technique Step 12: Researcher examines different outputs and their effects on child
Alternative Flow	Step 10: YOLO responds according the created storytelling arc`s climax phase Step 11: YOLO stimulates divergent thinking using contrast creativity technique

Exception Flow	-
Postconditions	Researcher get different result according to different YOLO softwares

Table 12: Use YOLO`s API to study child-robot interaction

Use case name	Develop software for YOLO
Actors	Developer, API
Description	Since YOLO software is open source, it is programmable and it can be used to see various output behaviors.
Data	Open source files of YOLO software
Preconditions	YOLO must be assembled and initialized
Stimulus	Developer creates a new source file using Python with the help of API
Basic Flow	Step 1: Developer writes a new source code to change creativity behavior Step 2: Developer changes the source code in the Raspberry-Pi System Step 3: Developer initializes YOLO's software programme. Step 4: Developer connects Raspberry-Pi System and the software programme via wi-fi
Alternative Flow #1	Step 1: Developer writes a new source code to change storytelling arc

Alternative Flow #2	Step 1: Developer writes a new source code to change social behavior
Exception Flow	-
Postconditions	After following these steps, YOLO is activated according to the newly created Python file.

Table 13: Develop software for YOLO

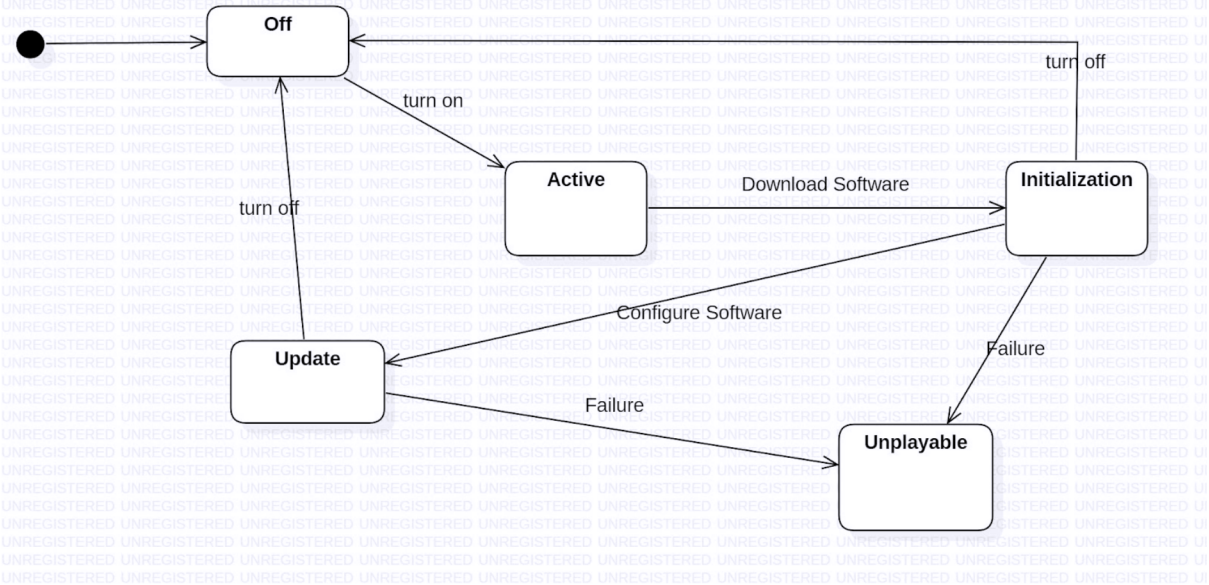


Figure 6: Statechart Diagram for Develop Software for YOLO

3.3 Usability Requirements

- Users shall be able to connect to YOLO user interface without internet connection via router.
- Users with proper internet connection shall be able to easily connect to YOLO API in less than 3 steps.
- Upon connection to the YOLO, users must be navigated to the User Interface for control over the YOLO.

- Registered users shall be able to use the API in order to make modifications on YOLO.
- Whenever a user wants to search for a setting in YOLO API, he/she shall be able to find it in at most 5 steps.
- Children and their parents shall use only one account to use YOLO API.
- All system users shall be able to use the API efficiently with basic computer or smartphone knowledge after seeing the quick introduction information.
- Users shall be able to view the current software of YOLO from API easily.
- Users shall be able to update and configure the software of YOLO from API easily.
- Whenever an error occurs, users shall be able to view detailed logs of errors from the API.
- Whenever a user wants to see the statistics of his/her or study groups' YOLO statistics, he/she shall be able to access detailed information about user statistics.
- Users shall be able to change the name of their YOLO according to their wishes.
- A user shall be able to use all related functions of the system wherever Internet connection is available.

3.4 Performance Requirements

- The response time of YOLO shall be less than 5 seconds to keep the user focused.
- Average response time of YOLO shall be less than or equal to 3 seconds for all user activities.
- Once the user connects to the user interface of YOLO, the time that it takes to see the login page shall not be longer than 3 seconds.
- Whenever a user wants to connect to the YOLO API, the time that it takes to show the user's YOLO's information page shall not exceed 5 seconds.
- Only 1 user shall be able to connect to the API at a time.
- Users shall be able to send only 1 package at a time.
- Speed of port connections between YOLO and router shall be faster than 25 mb/s.
- After a configuration is successfully done, the user shall be able to start using his/her YOLO in 2 minutes.
- After a software update, the user shall be able to start using his/her YOLO in 5 minutes.

3.5 Logical Database Requirements

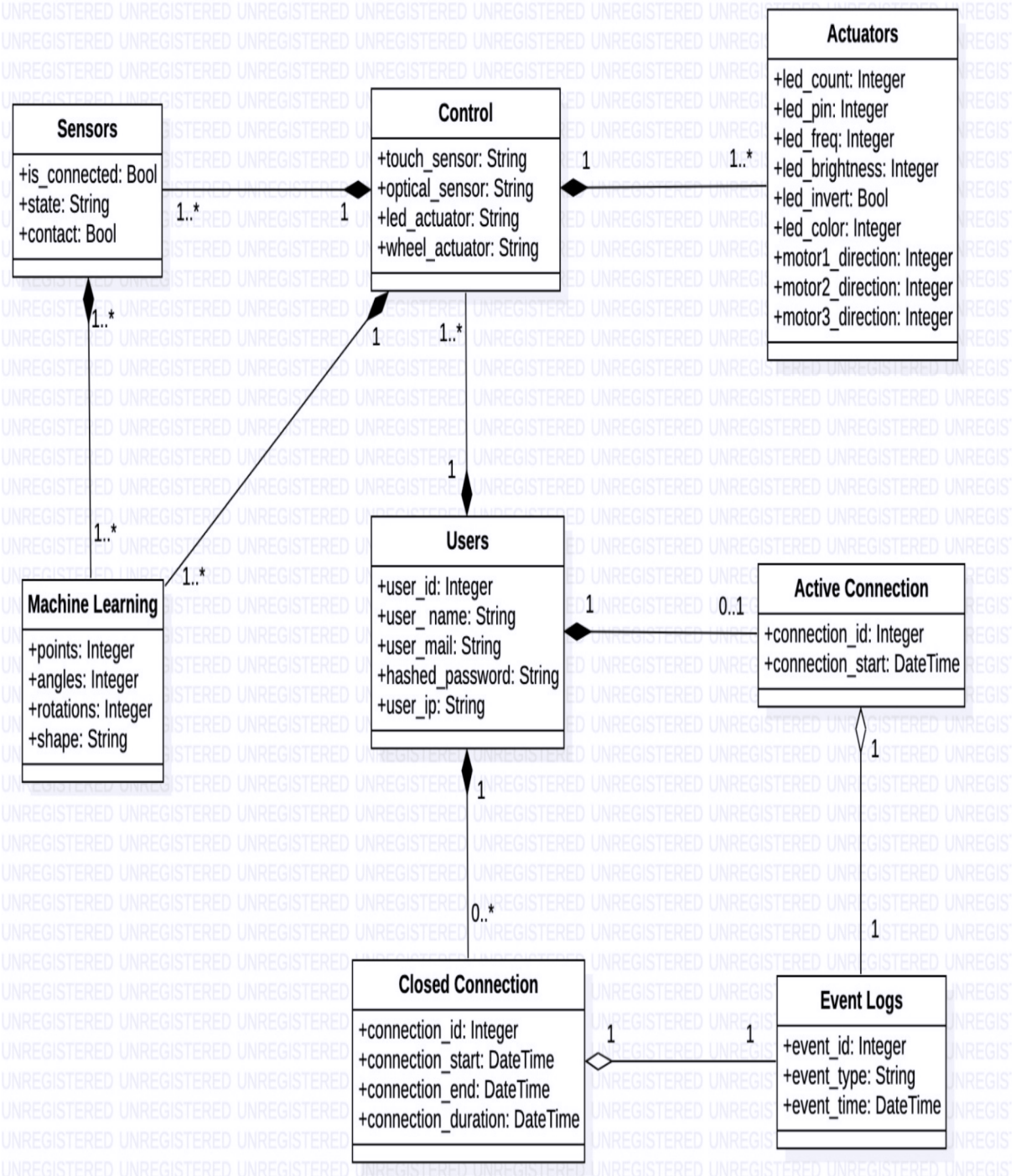


Figure 7: Logical Database Requirements Class Diagram

- Users shall interact with more than one control unit, but control units are not able to handle the interactions with more than one user.
- When a user connects to the YOLO API, a user entry and an active connection entry shall be created with unique IDs.
- Every machine learning task is born from sensor data and control unit decisions, there might be more than one (at least one) task and more than one (at least one) sensor that sends data for that task but there is only one control unit to decide what to do with machine learning output and sensor data.
- When a connection ends, the connection shall be saved by creating a closed connections entry, after that the entry in the active connections shall be deleted.
- Every active/closed connection has an exactly one event log table which stores performed events.
- One control might be responsible for more than one actuator but actuators are allowed to communicate with only one control unit.
- Every event performed by a user connection shall be saved by creating an event log entry.
- There might be more than one sensor that sends data to the control unit but the control unit needs at least one sensor to decide YOLO's actions. Moreover, sensors are connected to only one control unit.
- Users can have zero or more closed connections, but they can have at most one active connection.
- Upon deleting an user, all the active and closed connections which started by s/he shall be deleted.

3.6 Design Constraints

System concerns to serve users in a cheap and free way therefore open source hardware designs and open source software development methods are chosen and licensed under Creative Commons Attribution 4.0 International License.

3.7 Software System Attributes

a) Reliability

- All of the hardware and software code of the system must be open-source.
- In case of a connection failure or an unexpected shutdown, the system will store a detailed error log.
- In case of a sensor failure, other sensors shall continue their operations as if there is no data loss for machine learning activities.

b) Availability

- In case of a system restart, the whole system shall be available in less than 1 minute.
- In the absence of an internet connection, API still must be available via manual connection.
- Latest stable version of the system will be always available for users however it doesn’t update automatically.

c) Security

- The system components shall be tested regularly to avoid any attack or unwanted behavior.
- Whenever a new functionality is added, application logic tests shall be performed to avoid broken access controls and insecure direct object reference vulnerabilities.
- Remote access to the system's development environment shall be read-only.
- YOLO can be controlled by anyone who knows the IP address of its Raspberry Pi.

d) Maintainability

- Integration of new sensors shall not cause an error.
- Documentation of the system shall be found on project website
- Documentation of the system shall be updated regularly and shall help to use the system.

- Documentation of each development or bug fixing process shall be done in parallel with the procedure and international norms shall be followed.
- YOLO software must be able to be updated via the terminal on the server at any time.

e) Portability

- Users shall be able to connect to YOLO and use all related functions of the system via IP networks whenever there is an internet connection.
- Users shall be able to connect locally to YOLO and use all related functions using the peripheral I/O ports of the embedded Raspberry Pi.

3.8 Supporting Information

YOLO is an open-source project and its first aim is to help children toı improve their creativity. A user (parent or child him/herself) can find the necessary hardware components from stores or use a 3D printer in order to print the components and assemble them with the embedded software (Raspberry Pi) with the help of documentation provided. Moreover users with technical background or users who want to improve themselves on software development and Raspberry Pi coding can contribute to the project as an open-source developer.

4 Suggestions

YOLO is capable of interacting with children in a physical manner but when it comes to watching visual images, it fails. In order for YOLO to be more helpful for children, it needs to have a camera attached to it and with that camera, YOLO will be able to observe the mimics and gestures of users. Moreover, with upgraded software which can support image processing and better machine learning activities; YOLO might be a better friend for children because of its upgraded capabilities.