Software Requirements Specification
YOLO Robot

Andaç Berkay Seval 2235521

Musa Furkan Zenbilci 2469203

Table Of Contents

List of Figures

List of Tables

Revision History

Version	Date
1.0	07.04.2022
1.1	14.04.2022

Table 1: Change History

1 Introduction

1.1 Purpose of the System

YOLO (Your Own Living Object) is a non-anthropomorphic social robot designed and developed to stimulate creativity in children through storytelling activities. It was developed under a human-centered design approach with participatory design practices for two years and involving 142 children as active design contributors at all design stages. The purpose of this project is to help children to develop their two main creative thinking elements divergent and convergent thinking, aid them to give voice to their creative expressions and prevent "creative crisis" which is a decline in children's creativity abilities around the age of 7-9 years old.

1.2 Scope

In the scope of this system, children, their parents, and researchers from psychology area with different use cases are able to obtain a 3D-printed, open source, easy to use, behave socially intelligent, able to learn movement sets with artificial intelligence software and a low-cost than existing ones, creative stimulus robot. The robot is designed with low floor and wide walls principles to enable creativity provocations, open-ended playfulness, and disappointment avoidance through abstraction. The final product is a social robot designed for and with children.

The embedded hardware components along with installed system and artificial intelligence software enable to behave appropriate for the social situation, story-telling arc and understand the movement sets with an optical sensor, stimulate convergent and divergent thinking when necessary.

The robot is used for child playing, school teaching, academic research for psychology and more. Having customized operations enable robot to have wide range of usage possibilities.

Scope of the system could be listed as:

- The system shall be assembled and configured by the local personnel.
- The system shall use a Desktop API (application programming interface) to enable giving instructions and commands to the system.
- The system shall contain an embedded computer Raspberry Pi to operate system functions and artificial intelligence modules (KNN algorithm).
- The system shall allow client applications run on the embedded computer.
- Users shall be able to update the software of the robot without any need of hardware change.
- Users shall be able to develop additional functionality and entirely new movement sets, led light configurations and social behaviors without having to re-implement the more complex instrument control code.
- Users shall be able to install pre-existed open access functionality software to the system.

The robot does not have a camera and voice recognition module. Thus, functionalities like object detection, face recognition, natural language processing are not in the scope of the system.

1.3 System Overview

1.3.1 System Perspective

YOLO Robot is not a part of a larger system. However, it interacts with external systems such as Desktop API. With an application programming interface, users have ability to send instructions and commands to the YOLO, add new movement sets to YOLO, add new led configurations to YOLO, create new behaviors for a particular social situation for YOLO, change existing movement sets, led configurations and behaviors for YOLO. Also, users have ability to install pre-existed behavior models from internet to YOLO. In order to use desktop API, users shall connect YOLO to a computer via wi-fi or a USB cable. After the connection, users have ability to experiment with Raspberry Pi and give commands via API to YOLO with Python programming language. According to the playtime, movement shapes and touch information, YOLO will behave appropriately for the situation like mirror, contrast, puppeteer or randomly.

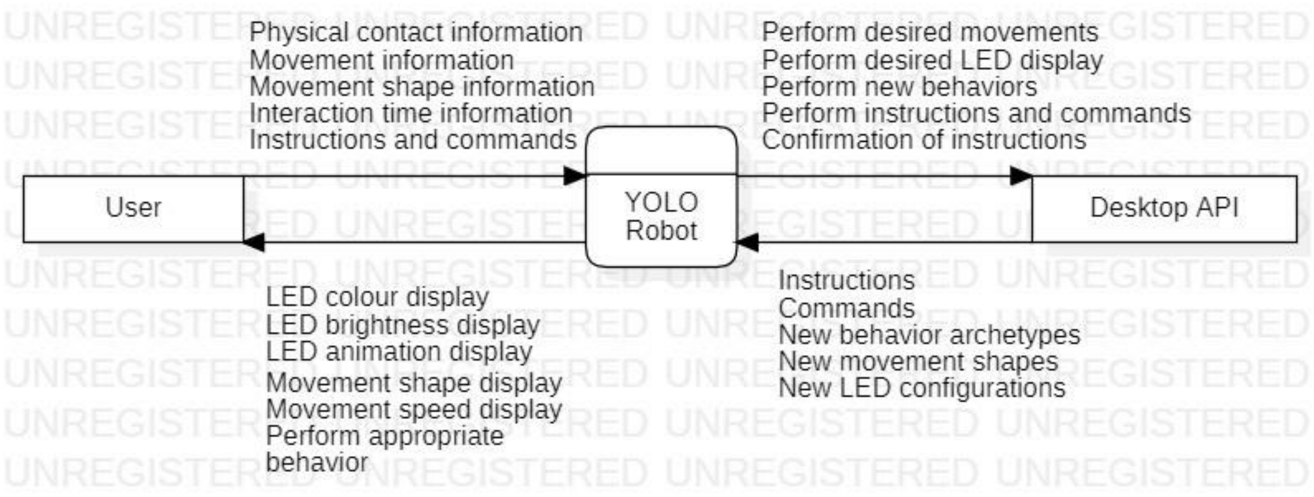


Figure 1: Context Diagram

1.3.1.1 System Interfaces

YOLO Robot is formed by the combination of hardware and software parts. These parts intercommunicate among themselves with signals and messages via hardware interfaces, software interfaces, communication interfaces and service interfaces.

- **Hardware Interfaces:** The embedded computer Raspberry Pi handles the interaction between all physical components of YOLO. According to signals from touch sensor, YOLO shall decide whether moving or show puppeteer behavior (lock itself to prevent movement). According to signals from optical sensor, YOLO shall determine the shape of the movement and behave appropriately for the situation. Wheel actuator provides navigation to YOLO for the movement with the wheels and adjust the speed for YOLO. LED actuator provides lights, brightness, and animations to YOLO. If there is an error in the hardware parts of the system, an error message shall be seen in the graphical user interface of the Raspberry Pi via connecting it with a computer.

- **Software Interfaces:** YOLO system executes on Raspberry Pi operating system. It includes a database management system to hold information about move sets, LED configurations and behavior types. Also, there are software packages that directly work with touch sensors and optical sensors to act appropriately. Also, it includes artificial intelligence software (pre-trained KNN model) to determine movement shapes using optical movement sensors. Furthermore, YOLO supports an API for giving direct commands, instructions and adding or changing movement, led light and behavior configurations. If there is an error in the software parts of the system, an error message shall be seen in the graphical user interface of the Raspberry Pi via connecting it with a computer.
- **Communication Interfaces:** YOLO system may connect to a computer with wireless communication or with a USB cable. Also, for direct installing of new movement sets, led configurations and behaviors, YOLO system may connect to internet with wi-fi or and ethernet cable directly. If there is an error in the communication parts of the system, an error message shall be seen in the graphical user interface of the Raspberry Pi via connecting it with a computer.

1.3.1.2 User Interfaces

Users can give instructions and commands to YOLO system with an API (application programming interface) after connecting YOLO to a computer via wi-fi or a USB cable. Users can manage the data in the YOLO, open a terminal to give commands, add new movement sets or change pre-existing ones, add new LED configurations, or change pre-existing ones and add new behaviors or change pre-existing ones, and install new behavior sets to YOLO with a graphical user interface which is offered by the Raspberry Pi embedded computer.

The first interface is the desktop display of the Raspbian Stretch Lite operating system running on the Raspberry Pi embedded computer. With this graphical user interface, users can open a terminal and give instructions and new commands to YOLO with Python programming language and add or change all functionalities of YOLO Robot.

Users, specifically children, provide input to YOLO Robot with playing them. YOLO Robot behave according to play type and raw inputs from the external world. Predefined software of YOLO Robot helps the creative process of playtime. According to movement shapes that YOLO Robot is played, with trained KNN algorithm, YOLO Robot understand the shape and according to story arc of the children, mirror the shape with convergent thinking process or move in a contrast way with divergent thinking. Also, according to raw inputs provided by the user, YOLO Robot decides its LED configuration, light animations, brightness, and color palette. Furthermore, YOLO Robot could decide its behavior type according to touching time, movement shapes and playtime.

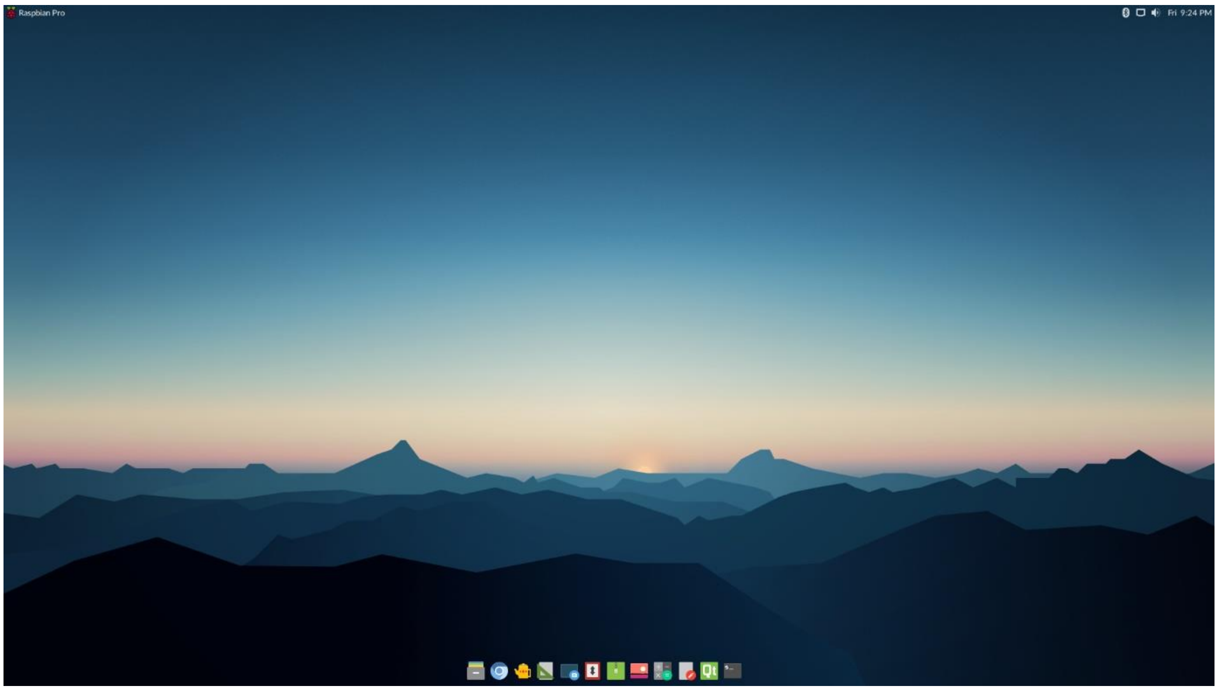
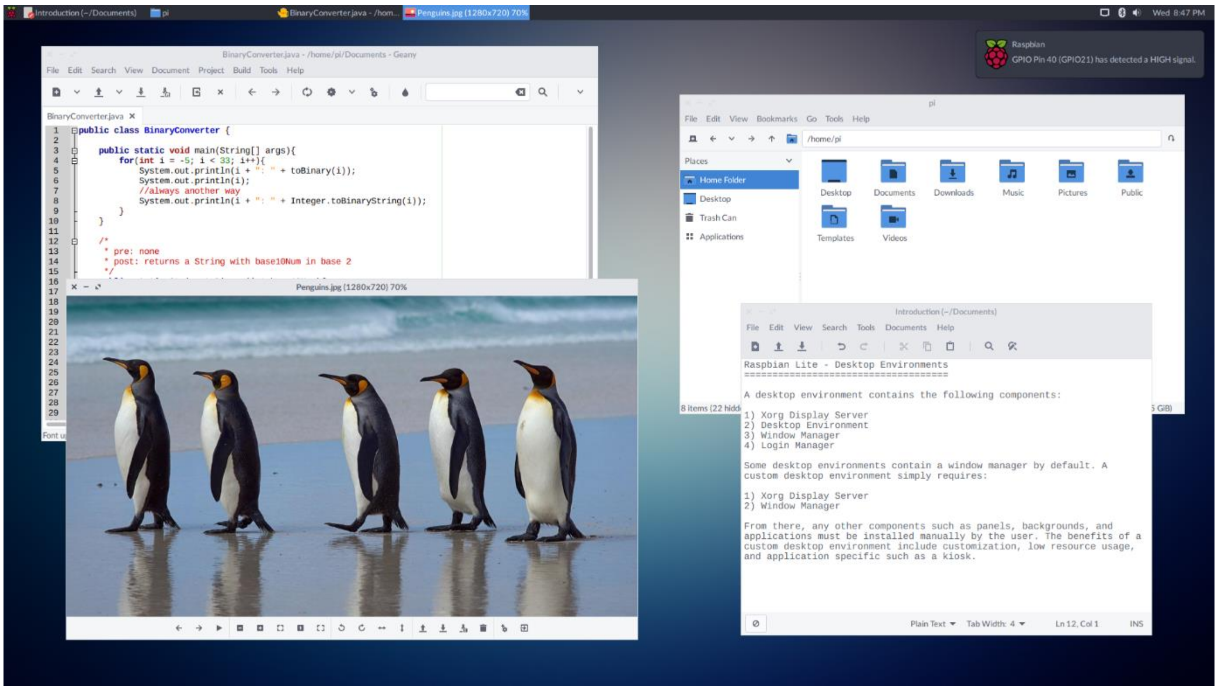


Figure 2: Raspbian Lite Desktop (YOLO’S OS)


Introduction (~/Documents)
pi
BinaryConverter.java - /hom...
Penguins.jpg (1280x720) 70%
Wed 8:47 PM
BinaryConverter.java - /home/pi/Documents - Geany
File Edit Search View Document Project Build Tools Help
BinaryConverter.java
Penguins.jpg (1280x720) 70%
pi
File Edit View Bookmarks Go Tools Help
/home/pi
Places
Home Folder
Desktop
Trash Can
Applications
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
Videos
Introduction (~/Documents)
File Edit View Search Tools Documents Help
Plain Text
Tab Width: 4
Ln 12, Col 1
INS

Raspbian
GPIO Pin 40 (GPIO21) has detected a HIGH signal.

```
public class BinaryConverter {

    public static void main(String[] args){
        for(int i = -5; i < 33; i++){
            System.out.println(i + ": " + toBinary(i));
            System.out.println(i);
            //always another way
            System.out.println(i + ": " + Integer.toBinaryString(i));
        }
    }

    /*
     * pre: none
     * post: returns a String with base10Num in base 2
     */
```

```
Raspbian Lite - Desktop Environments
====================================

A desktop environment contains the following components:

1) Xorg Display Server
2) Desktop Environment
3) Window Manager
4) Login Manager

Some desktop environments contain a window manager by default. A
custom desktop environment simply requires:

1) Xorg Display Server
2) Window Manager

From there, any other components such as panels, backgrounds, and
applications must be installed manually by the user. The benefits of a
custom desktop environment include customization, low resource usage,
and application specific such as a kiosk.
```

Figure 3: Raspbian Pi Folders and Programming Tools (YOLO’S OS)

```
jimmy@RPI3B:~ $ free -m
              total        used        free      shared  buff/cache   available
Mem:            927         146         590          12         190         718
Swap:            99           0          99
jimmy@RPI3B:~ $ 
```

Figure 4: Raspbian Pi Terminal (YOLO’S OS)

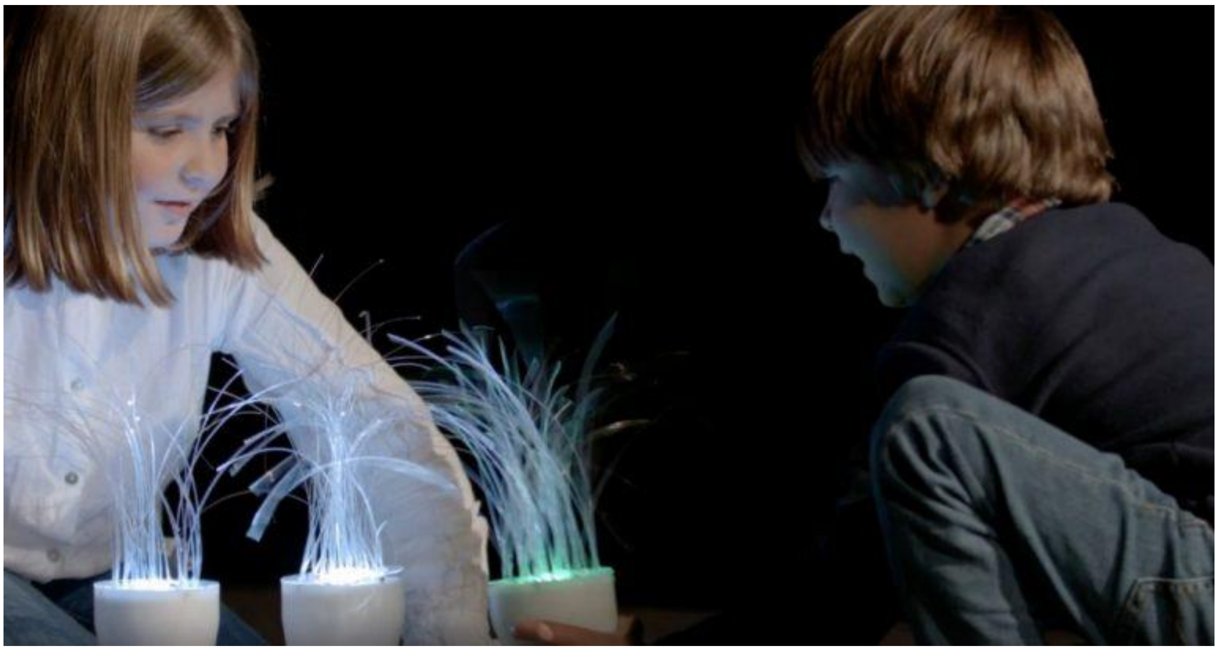


Figure 5: Child – YOLO interaction

1.3.2 System Functions

Function	**Description**
Open YOLO Robot	User opens YOLO Robot with pressing open button on the robot.
Connect YOLO Robot to a computer	User connects YOLO Robot to a laptop or a desktop with wi-fi or with a USB cable to get graphical user interface.
Give instructions to YOLO Robot	User gives instructions and commands to YOLO Robot with a graphical user interface terminal after the connection to a computer.
Change move configurations of YOLO Robot	User changes move configurations of YOLO Robot with giving instructions.
Change LED light options of YOLO Robot	User changes LED light options of YOLO Robot with giving instructions.
Change preexisted behavior options of YOLO Robot	User changes preexisted behavior options of YOLO Robot with giving instructions.
Add new move types to YOLO Robot	User adds new move types to YOLO Robot with providing new move type codes.
Add new LED light adjustments to YOLO Robot	User adds new LED light adjustments to YOLO Robot with providing LED configuration codes.
Add new behavior to YOLO Robot	User adds new behavior to YOLO Robot with providing behavior type code.
Play with YOLO Robot	User plays with YOLO Robot in a desired way like touching, untouching, moving, drawing shapes on the floor or like an ordinary toy.
Close YOLO Robot	User closes YOLO Robot with pressing close button on the robot.

Table 2: System Functions

1.3.3 Stakeholder Characteristics

There are different group of users within the system.

- **Users (Children):** Users are the people who play with YOLO Robot with different aspects. They can improve their creative skills by playing with YOLO. They can play with YOLO in a desired way like touching, not touching, moving, drawing shapes on the ground or like an ordinary toy. According to playstyle, YOLO Robot responds differently to the certain situation.
- **Other Users (Teacher, Parents, Researchers):** There can be multiple other users who use the YOLO system. Teachers and parents can help children to give instructions to YOLO Robot, Also, they can crate new storylines for children to improve their thinking capacity in an educational manner. Researcher can change move configurations, LED light options, preexisted behavior options or add new move types, new LED light adjustments, new behavior to YOLO Robot to do experiments with the system with participating children.

- **IT People:** Administrators and maintainers shall have strong technical skills and knowledge to maintain the YOLO's software system and keep the system open for functional extensions such as new move types, LED light options and behavior types.

1.3.4 Limitations

- **Regulatory policies:** YOLO Robot is an open-source hardware and software project. Thus, STL format design files and project codes are accessible to everyone.
- **Hardware limitations:** YOLO Robot uses a Raspberry Pi embedded computer to run the system. Touch sensors and optical sensors shall work properly, and response time shall not be more than 200ms to work in a desired way.
- **Interfaces to other applications:** YOLO Robot system shall be compatible with different hardware and different operating systems to connect.
- **Parallel operation:** YOLO Robot system shall be able to process different tasks concurrently, such as moving in a desired way, adjusting LED color, animations, and brightness, and behaving according to situation.
- **Audit functions:** There is no such limitation since it is an open-source system.
- **Control functions:** YOLO Robot system is not controlled by a center but by a community.
- **Higher-order language requirements:** Raspberry Pi runs Python code to control physical hardware and software of the system. Also, desktop API supports Python code.
- **Signal handshake protocols:** There is no such limitation since it is a local system, remote control is not supported.
- **Quality requirements:** Stability, maintainability and ease-of-usage are important priorities for YOLO Robot system. Therefore, YOLO's hardware and software shall be easily built, installed, and used globally.
- **Criticality of the application:** YOLO Robot system is not a critical system. In case of any errors, IT people can solve the problem.
- **Safety and security considerations:** Due to the material used for YOLO such as battery, high temperature objects shall not be near of the system.
- **Physical/mental considerations:** Physically/mentally disabled people can user YOLO Robot with the help of someone.
- **Limitations that are sourced from other systems:** In order to connect YOLO to a computer, there shall be a wi-fi connection or computer shall have a USB port.

1.4 Definitions

Term	**Definition**
API	Application Programming Interface
IT People	Information Technology People
GUI	Graphical User Interface
KNN Algorithm	K-Nearest Neighbors Algorithm
AI	Artificial Intelligence
STL Format	Standard Triangle Language Format

Table 3: Definitions

2 References

This document is prepared with respect to IEEE 29148-2018 standard:

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

Other sources:

Oliveira P.A., Gomes S., Chandak A., Arriaga P., Hoffman G., Paiva A. (February 05, 2020). *Software architecture for YOLO, a creativity-stimulating robot*. Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY, USA

Oliveira P.A., Paiva A., Hoffman G., Arriaga P. (March 8, 2021). *Children as Robot Designers*. University of Washington Seattle, WA, USA

Oliveira P.A., Arriaga P., Paiva A., Hoffman G. (July 14, 2019). *Guide to build YOLO, a creativity-stimulating robot for children*. ISCTRE-Instituto Universitario de Lisboa, CIS-IUL, Lisbon, Portugal

3 Specific Requirements

3.1 External Interfaces

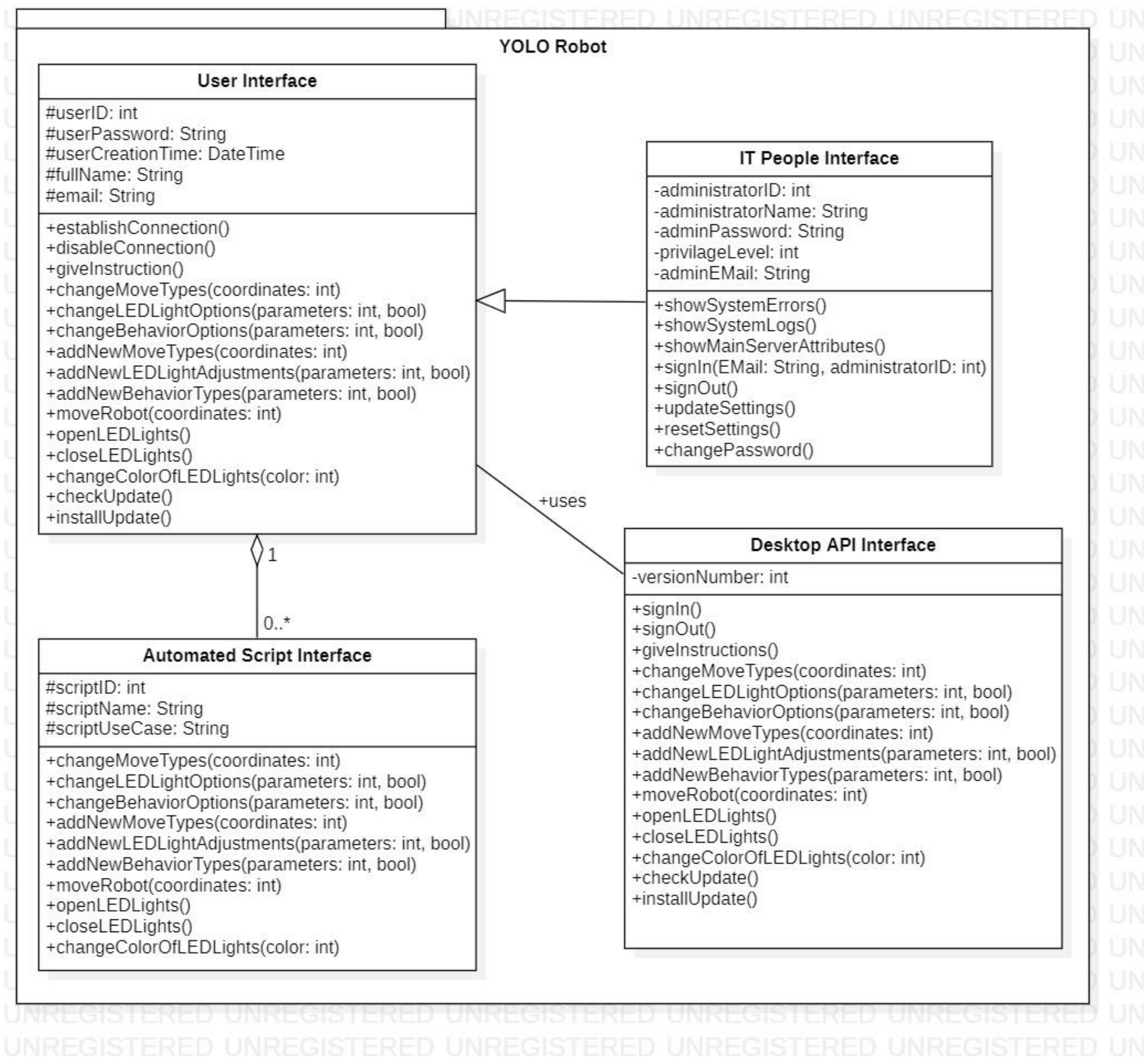

YOLO Robot
User Interface
#userID: int
#userPassword: String
#userCreationTime: DateTime
#fullName: String
#email: String
+establishConnection()
+disableConnection()
+giveInstruction()
+changeMoveTypes(coordinates: int)
+changeLEDLightOptions(parameters: int, bool)
+changeBehaviorOptions(parameters: int, bool)
+addNewMoveTypes(coordinates: int)
+addNewLEDLightAdjustments(parameters: int, bool)
+addNewBehaviorTypes(parameters: int, bool)
+moveRobot(coordinates: int)
+openLEDLights()
+closeLEDLights()
+changeColorOfLEDLights(color: int)
+checkUpdate()
+installUpdate()
IT People Interface
-administratorID: int
-administratorName: String
-adminPassword: String
-privilageLevel: int
-adminEMail: String
+showSystemErrors()
+showSystemLogs()
+showMainServerAttributes()
+signIn(EMail: String, administratorID: int)
+signOut()
+updateSettings()
+resetSettings()
+changePassword()
+uses
1
0..*
Automated Script Interface
#scriptID: int
#scriptName: String
#scriptUseCase: String
+changeMoveTypes(coordinates: int)
+changeLEDLightOptions(parameters: int, bool)
+changeBehaviorOptions(parameters: int, bool)
+addNewMoveTypes(coordinates: int)
+addNewLEDLightAdjustments(parameters: int, bool)
+addNewBehaviorTypes(parameters: int, bool)
+moveRobot(coordinates: int)
+openLEDLights()
+closeLEDLights()
+changeColorOfLEDLights(color: int)
Desktop API Interface
-versionNumber: int
+signIn()
+signOut()
+giveInstructions()
+changeMoveTypes(coordinates: int)
+changeLEDLightOptions(parameters: int, bool)
+changeBehaviorOptions(parameters: int, bool)
+addNewMoveTypes(coordinates: int)
+addNewLEDLightAdjustments(parameters: int, bool)
+addNewBehaviorTypes(parameters: int, bool)
+moveRobot(coordinates: int)
+openLEDLights()
+closeLEDLights()
+changeColorOfLEDLights(color: int)
+checkUpdate()
+installUpdate()

Figure 6: External Interfaces Class Diagram

User Interface

- Users have a userID, and all related information saved in the system.
- Users choose to connect to a computer and disconnect.
- Users can give instructions, change move types, change LED light options, change behavior options, and add new move types, new LED light adjustments, new behavior types by using desktop API.
- Users can also give these instructions and commands with giving an automated script to the system.
- Users can use automated scripts as much as they want. They do not have to use them.

Automated Script Interface:

- Automated Scripts have unique IDs with their use cases.
- Automated Scripts are written by users, and they can give instructions, change and add configurations to the system.
- Users can use automated scripts as much as they want. They do not have to use them.

IT People Interface:

- IT People are administrators and maintainers of the system.
- They have unique IDs and passwords which are saved in the system database.
- They can see system errors, system logs.
- They can update system settings. Also, they can reset system settings.

Desktop API Interface:

- Desktop API has a version number which indicates its latest update number.
- Users can give instructions, change move types, change LED light options, change behavior options, and add new move types, new LED light adjustments, new behavior types by using desktop API.
- Desktop API can be updated with checking and installing updates.

3.2 Functions

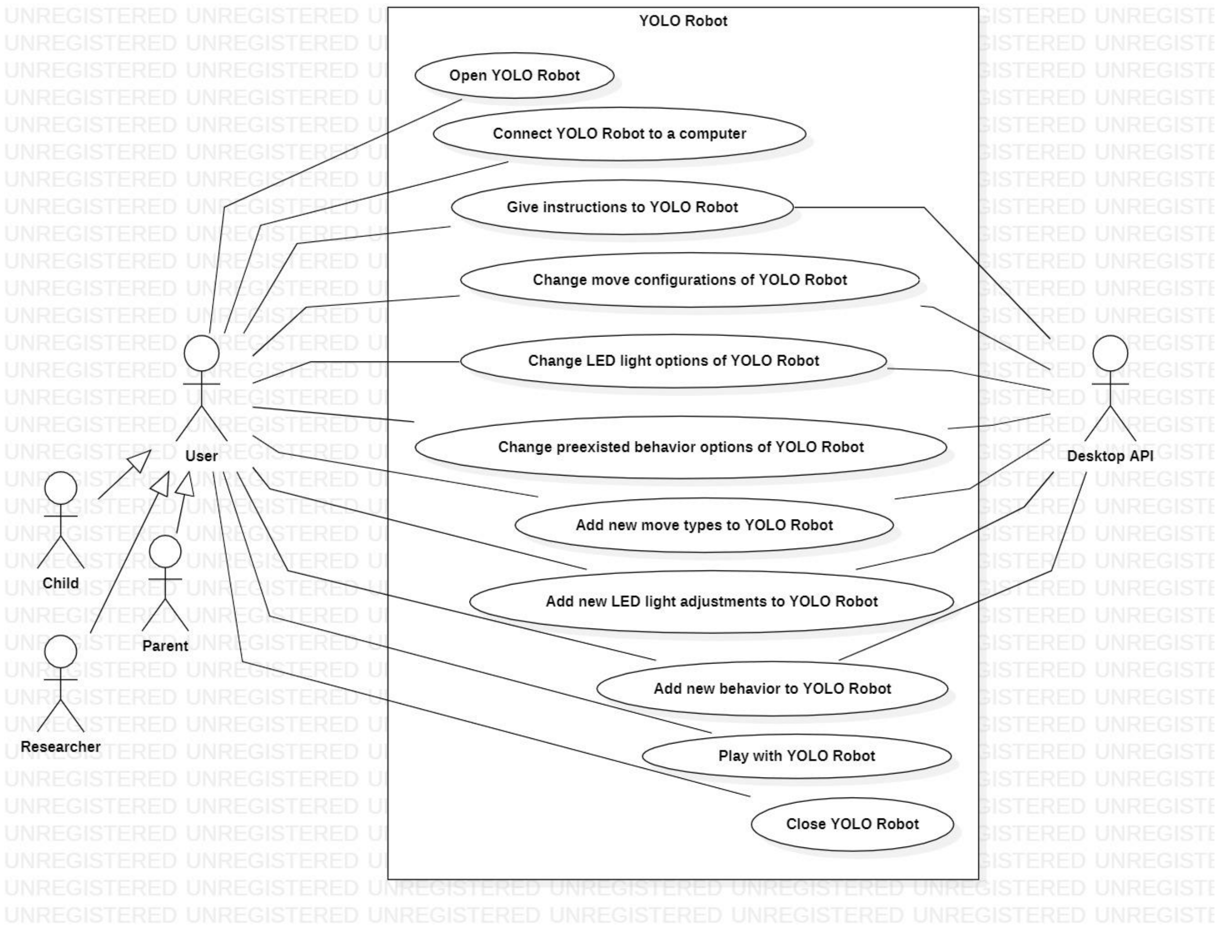

YOLO Robot
Open YOLO Robot
Connect YOLO Robot to a computer
Give instructions to YOLO Robot
Change move configurations of YOLO Robot
Change LED light options of YOLO Robot
Change preexisted behavior options of YOLO Robot
Add new move types to YOLO Robot
Add new LED light adjustments to YOLO Robot
Add new behavior to YOLO Robot
Play with YOLO Robot
Close YOLO Robot
User
Child
Parent
Researcher
Desktop API

Figure 7: Use Case Diagram

Use Case Name	Open YOLO Robot
Actors	User
Description	User opens YOLO Robot with pressing open button on the robot.
Data	Open switch from battery of YOLO Robot
Preconditions	Battery of the YOLO Robot shall not be empty in order to provide electricity to system.
Stimulus	User presses open button.
Basic Flow	Step 1: User presses open button of the YOLO Robot Step 2: YOLO Robot starts giving responses
Exception Flow	Step 2: If battery of the YOLO Robot is empty, user shall charge it
Postconditions	After YOLO Robot opens, user could play with it.

Table 4: Open YOLO Robot

Use Case Name	Connect YOLO Robot to a computer
Actors	User
Description	User connects YOLO Robot to a laptop or a desktop with wi-fi or with a USB cable to get graphical user interface.
Data	Wi-fi signals or data on the USB
Preconditions	For wi-fi connection, computer shall provide a wi-fi connection. For cable connection, computer shall provide a USB cable port.
Stimulus	Users connects the system to a computer with wi-fi or a USB cable.
Basic Flow	For wi-fi connection: Step 1: User presses wi-fi button on the YOLO Robot Step 2: User does necessary adjustment on computer and connects YOLO Robot to computer For USB cable connection: Step 1: User sticks one side of a USB cable to YOLO Robot Step2: User sticks other side of the USB cable to a computer and connects the system
Exception Flow	For wi-fi connection: Step 2: If connection is weak, user shall change the location of the system to a closer place to wi-fi For USB cable connection: Step 2: If cable is broken or USB port is broken on the computer, user shall the cable or change the port
Postconditions	Connection is successful and ready for commands.

Table 5: Connect YOLO Robot to a computer

Use Case Name	Give instructions to YOLO Robot
Actors	User, Desktop API
Description	User gives instructions and commands to YOLO Robot with a graphical user interface terminal after the connection to a computer.
Data	Python code, commands
Preconditions	Connection between YOLO Robot and computer shall be successful and graphical user interface shall be responsive.
Stimulus	User gives commands and instructions from terminal to YOLO Robot.
Basic Flow	Step 1: User writes python code in terminal for giving instructions Step 2: User sends the code to YOLO Step 3: YOLO behaves according to commands
Exception Flow	Step 1: If user gives erroneous code (syntax or semantics mistake), user shall change the code and instructions
Postconditions	YOLO Robot does what is wanted.

Table 6: Give instructions to YOLO Robot

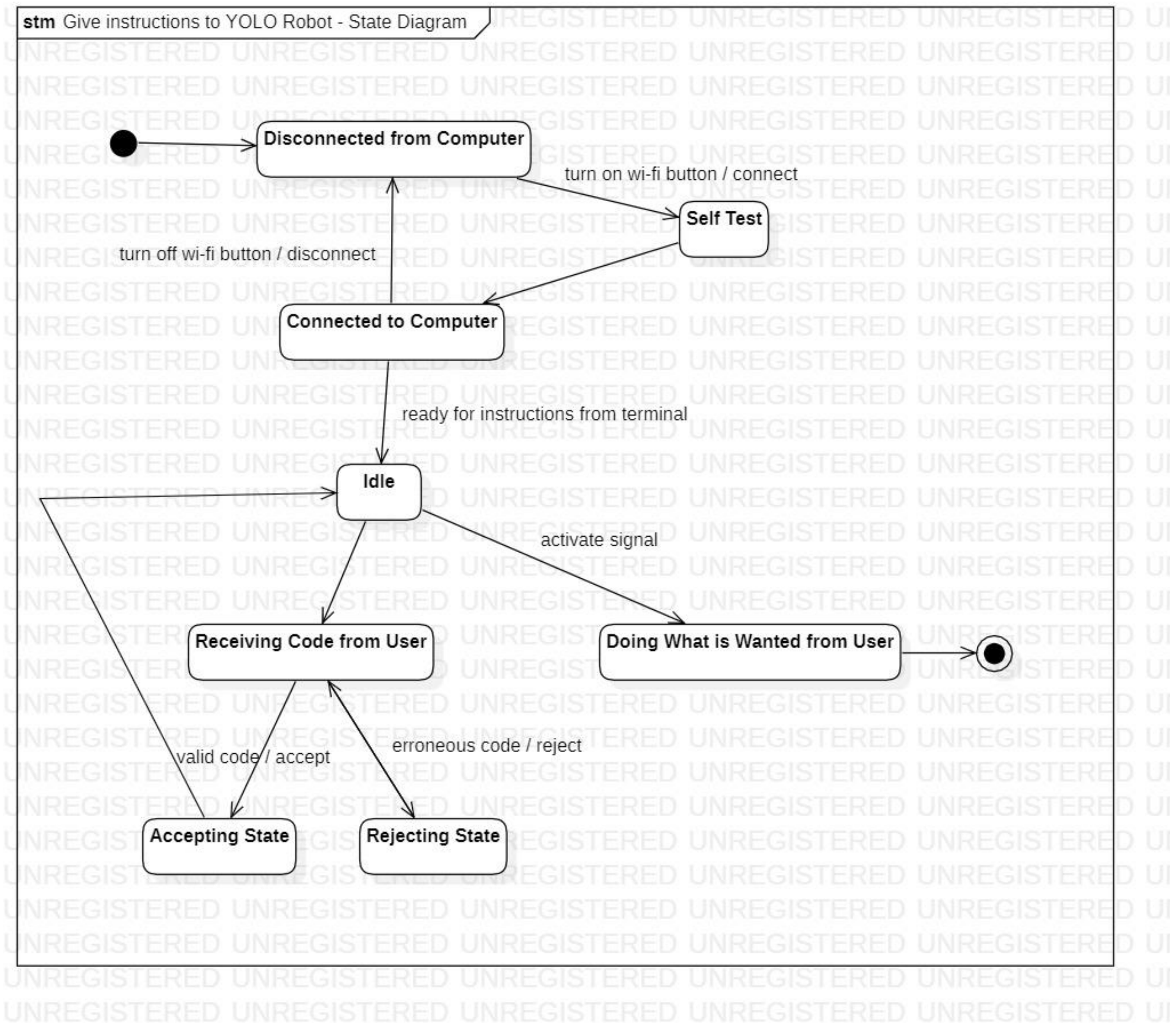

stm Give instructions to YOLO Robot - State Diagram
Disconnected from Computer
turn on wi-fi button / connect
Self Test
turn off wi-fi button / disconnect
Connected to Computer
ready for instructions from terminal
Idle
activate signal
Receiving Code from User
Doing What is Wanted from User
valid code / accept
erroneous code / reject
Accepting State
Rejecting State

Figure 8: State Diagram of Give instructions to YOLO Robot

Use Case Name	Change move configurations of YOLO Robot
Actors	User, Desktop API
Description	User changes move configurations of YOLO Robot with giving instructions.
Data	Python code, commands
Preconditions	Connection between YOLO Robot and computer shall be successful and graphical user interface shall be responsive.
Stimulus	User gives instructions to change move configurations of YOLO Robot.
Basic Flow	Step 1: User writes python code in terminal to change move configurations Step 2: User sends the code to YOLO Step 3: YOLO moves according to commands
Exception Flow	Step 1: If user gives erroneous code (syntax or semantics mistake), user shall change the code and instructions
Postconditions	YOLO Robot moves in the desired way.

Table 7: Change move configurations of YOLO Robot

Use Case Name	Change LED light options of YOLO Robot
Actors	User, Desktop API
Description	User changes LED light options of YOLO Robot with giving instructions.
Data	Python code, commands
Preconditions	Connection between YOLO Robot and computer shall be successful and graphical user interface shall be responsive.
Stimulus	User gives instructions to change LED light options of YOLO Robot.
Basic Flow	Step 1: User writes python code in terminal to change LED light options Step 2: User sends the code to YOLO Step 3: YOLO’s light color, brightness and light animations change according to commands
Exception Flow	Step 1: If user gives erroneous code (syntax or semantics mistake), user shall change the code and instructions
Postconditions	YOLO Robot shows light options in the desired way.

Table 8: Change LED light options of YOLO Robot

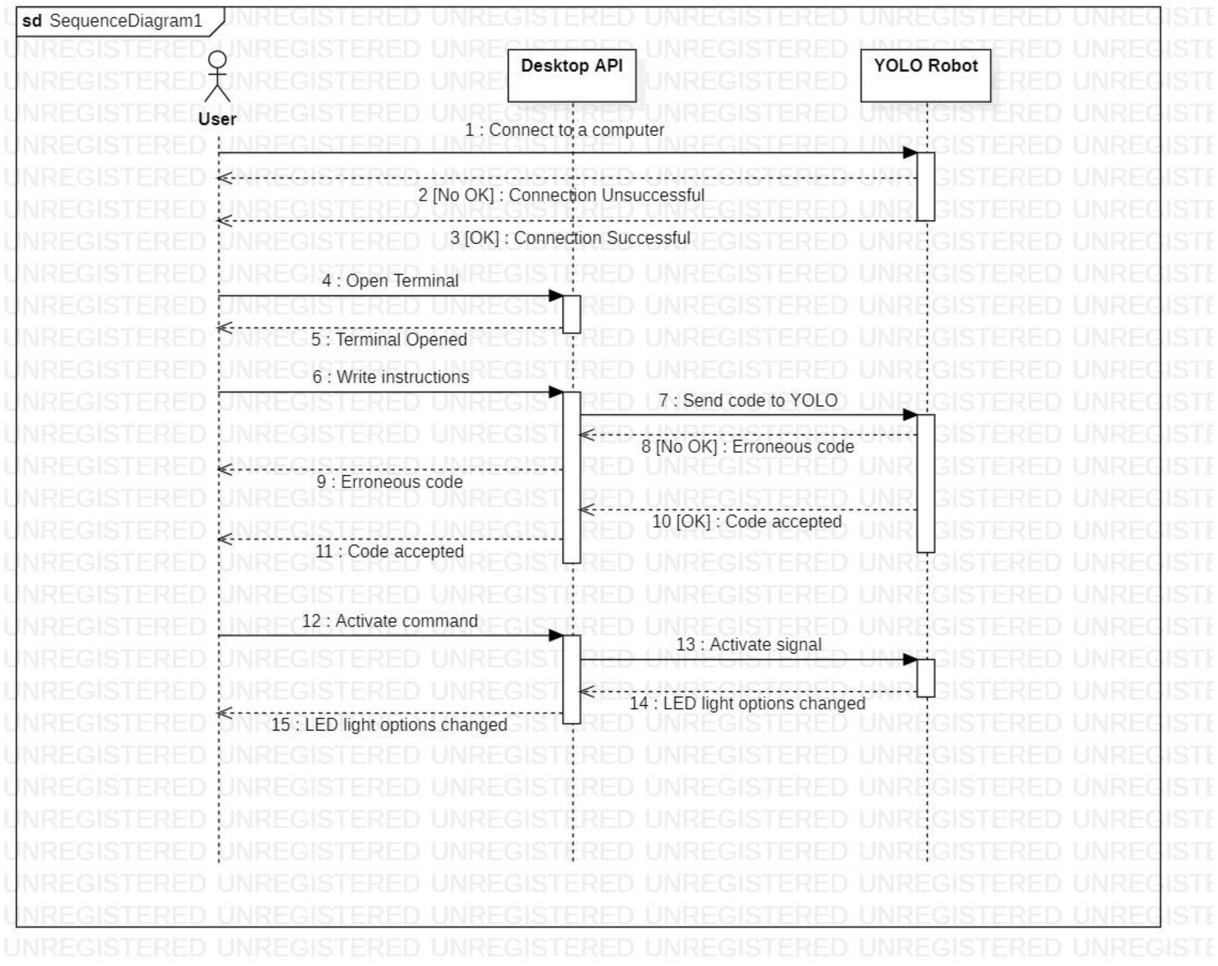


Figure 9: Sequence Diagram of Change LED light options of YOLO Robot

Use Case Name	Change preexisted behavior options of YOLO Robot
Actors	User, Desktop API
Description	User changes preexisted behavior options of YOLO Robot with giving instructions.
Data	Python code, commands
Preconditions	Connection between YOLO Robot and computer shall be successful and graphical user interface shall be responsive.
Stimulus	User gives instructions to change behavior options of YOLO Robot.
Basic Flow	Step 1: User writes python code in terminal to change behavior configurations Step 2: User sends the code to YOLO Step 3: YOLO behaves according to commands
Exception Flow	Step 1: If user gives erroneous code (syntax or semantics mistake), user shall change the code and instructions
Postconditions	YOLO Robot behaves in the desired way.

Table 9: Change preexisted behavior options of YOLO Robot

Use Case Name	Add new move types to YOLO Robot
Actors	User, Desktop API
Description	User adds new move types to YOLO Robot with providing new move type codes.
Data	Python code, commands
Preconditions	Connection between YOLO Robot and computer shall be successful and graphical user interface shall be responsive.
Stimulus	User provides python code to add new move types for YOLO Robot.
Basic Flow	Step 1: User writes python code or download preexisted code and provides in terminal to add new move types Step 2: User sends the code to YOLO Step 3: YOLO moves according to commands
Exception Flow	Step 1: If user gives erroneous code (syntax or semantics mistake), user shall change the code and instructions
Postconditions	YOLO Robot moves in the desired way.

Table 10: Add new move types to YOLO Robot

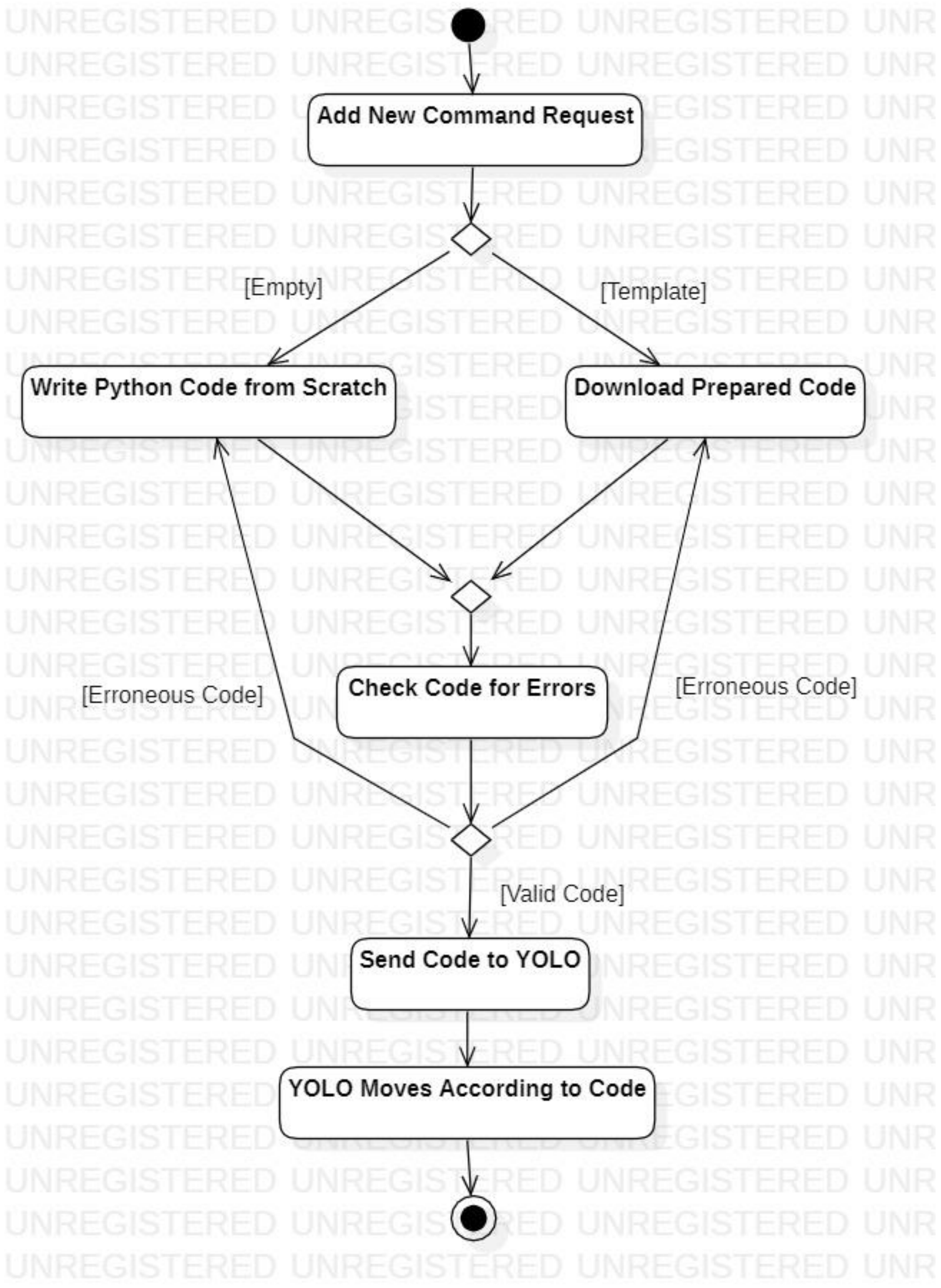


Figure 10: Activity Diagram of Add new move types to YOLO Robot

Use Case Name	Add new LED light adjustments to YOLO Robot
Actors	User, Desktop API
Description	User adds new LED light adjustments to YOLO Robot with providing LED configuration codes.
Data	Python code, commands
Preconditions	Connection between YOLO Robot and computer shall be successful and graphical user interface shall be responsive.
Stimulus	User provides python code to add new LED light options for YOLO Robot.
Basic Flow	Step 1: User writes python code or download preexisted code and provides in terminal to add new LED light adjustments Step 2: User sends the code to YOLO Step 3: YOLO's light color, brightness and light animations are updated according to given code
Exception Flow	Step 1: If user gives erroneous code (syntax or semantics mistake), user shall change the code and instructions
Postconditions	YOLO Robot shows light options in the desired way.

Table 11: Add new LED light adjustments to YOLO Robot

Use Case Name	Add new behavior to YOLO Robot
Actors	User, Desktop API
Description	User adds new behavior to YOLO Robot with providing behavior type code.
Data	Python code, commands
Preconditions	Connection between YOLO Robot and computer shall be successful and graphical user interface shall be responsive.
Stimulus	User provides python code to add new behavior for YOLO Robot.
Basic Flow	Step 1: User writes python code or download preexisted code and provides in terminal to add new behavior Step 2: User sends the code to YOLO Step 3: YOLO behaves according to given code
Exception Flow	Step 1: If user gives erroneous code (syntax or semantics mistake), user shall change the code and instructions
Postconditions	YOLO Robot behaves in the desired way.

Table 12: Add new behavior to YOLO Robot

Use Case Name	Play with YOLO Robot
Actors	User
Description	User plays with YOLO Robot in a desired way like touching, untouching, moving, drawing shapes on the floor or like an ordinary toy.
Data	Touch information, 2D input coordinates (move information), playtime
Preconditions	Battery of the YOLO Robot shall not be empty in order to provide electricity to system.
Stimulus	User play with YOLO Robot.
Basic Flow	Step 1: User touches the YOLO Robot Step 2: User moves the YOLO Robot Step 3: YOLO Robot responds according to given inputs from user
Exception Flow	-
Postconditions	User plays with YOLO Robot, and it behaves according to user.

Table 13: Play with YOLO Robot

Use Case Name	Close YOLO Robot
Actors	User
Description	User closes YOLO Robot with pressing close button on the robot.
Data	Close switch from battery of YOLO Robot
Preconditions	YOLO Robot shall be opened.
Stimulus	User presses close button.
Basic Flow	Step 1: User presses close button of the YOLO Robot Step 2: YOLO Robot stops giving responses (no electricity)
Exception Flow	-
Postconditions	YOLO Robot is closed.

Table 14: Close YOLO Robot

3.3 Usability Requirements

- Users shall be able to connect the YOLO Robot to a computer with wi-fi or with a USB cable.
- Users shall be able to have a GUI (graphical user interface) for the YOLO Robot Raspberry Pi OS.
- Users shall be able to give instructions and commands to YOLO Robot directly from terminal.

- Users shall be able to change move configurations of YOLO Robot with Python code.
- Users shall be able to change LED light options of YOLO Robot.
- Users shall be able to change preexisted behavior options of YOLO Robot.
- Users shall be able to add new move types to YOLO Robot with Python code.
- Users shall be able to add new LED light adjustments to YOLO Robot.
- Users shall be able to add new behavior types to YOLO Robot with writing from scratch or downloading prepared Python code.
- Users shall be able play with YOLO Robot with touching, untouching, moving, drawing shapes on the floor or like an ordinary toy.

3.4 Performance Requirements

- Latency between touch sensors and behavior mechanism shall not exceed 200ms.
- Latency between optical sensors and wheel configuration mechanism shall not exceed 500ms.
- YOLO Robot system shall have a 2 GB RAM to store necessary information of move types, LED light options and behavior types.
- YOLO Robot system shall support wi-fi connection.
- YOLO Robot system shall have a USB port and support cable connection.
- In child-robot interaction, YOLO Robot’s response time shall not exceed 300ms.
- Signal time between hardware parts of YOLO Robot should not exceed 100ms.
- Wheels of YOLO Robot shall not support to have a speed of more than 1m/s to prevent environmental damage.

3.5 Logical Database Requirements

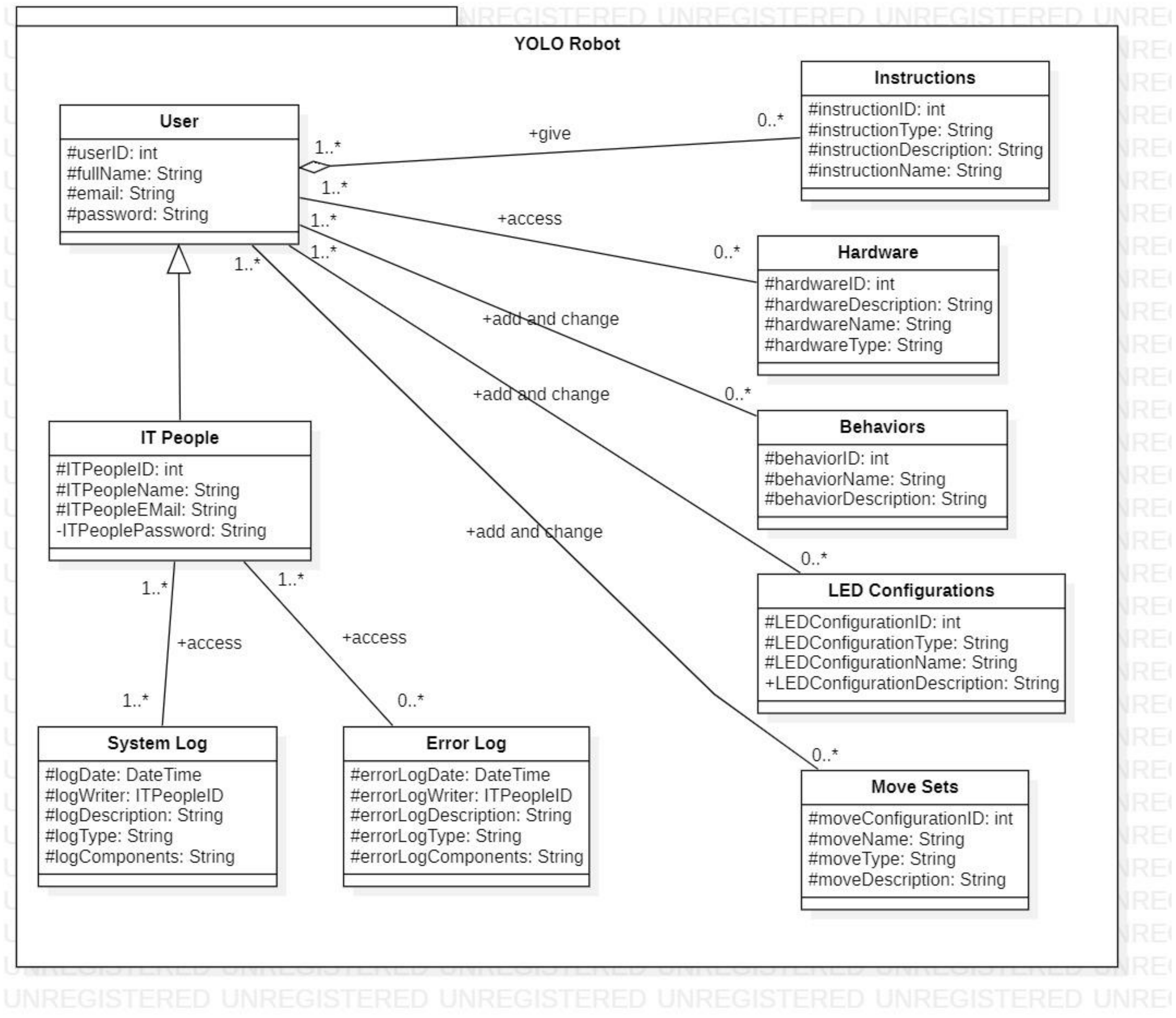


Figure 11: Logical Database Requirements Class Diagram

- Each user has a unique ID, e-mail and password in the system.
- IT People is a generalization class of users.
- IT People have unique IDs, e-mails and passwords.
- IT People can change their account information.
- Only IT People (administrators and maintainers) can access System Logs and Error Logs.
- System Logs shall be taken each day. They can be taken by one or more IT People.
- In case of any error happening, Error Log shall be taken by one or more IT People.
- If there are no errors, no Error Log shall be taken.
- Users give instructions to the system.
- There is an aggregation relationship between users and instructions since in case of no users, instructions can still be entities.
- Each instruction has a unique ID and name.
- Users can give zero or more instructions to the system.
- Each hardware component has a unique ID and name.

- Users can access to hardware component with GUI while connecting to a computer.
- Users can add and change behavior types of the system.
- Each behavior type has a unique ID and name.
- Users can add and change zero or more behaviors.
- Users can add and change LED configurations of the system.
- Each LED configuration has a unique ID and name.
- Users can add and change zero or more LED configurations.
- Users can add and change move types of the system.
- Each move type has a unique ID and name.
- Users can add and change zero or more move types.

3.6 Design Constraints

- System shall protect behavior of YOLO Robot system by inspecting user created code snippets to avoid aggressive usage of the system.
- Due to the storage constraints of the YOLO Robot, only one user's installed code snippets can work at once.
- Since the essential users are children, YOLO Robot system will have move constraints to avoid possible injuries to end users or defects on the robot.

3.7 System Attributes

3.7.1 Reliability

- All hardware and software components of YOLO Robot shall be open source.
- YOLO Robot system shall support software extensions like new move types, LED light adjustments and behavior configurations.
- Failure time of the YOLO system's hardware shall be less than 5 minutes in a year.
- Delay between optical sensors and move actuators shall be less than 500ms.
- In case of connection to a computer, YOLO system shall be compatible with any operating system such as Windows, Linux and MacOS.
- YOLO system shall support shell instructions from terminal.

3.7.2 Availability

- In case of system restart, YOLO system shall be available within 2 minutes.
- In case of failure, YOLO system shall restart in 1.5 seconds.
- After restart, local storage of YOLO system shall be recovered and not lost.
- In case of updating Raspberry Pi OS, YOLO system shall be connected to a computer which is connected to Internet.

3.7.3 Security

- Whenever a new functionality (move types, LED configurations, behavior options) is added to the system, tests shall be done to avoid breaking the system.
- Internal database of the system shall keep log of the YOLO Robot system.
- The system components shall be tested regularly to avoid system hazards.

3.7.4 Maintainability

- YOLO system software shall be updated by connecting system to a computer.
- Integration of new functionalities shall not cause any harm to the system.
- Documentation of the system with STL files shall be found on project articles.
- YOLO Robot project is supported and maintained by community in GitHub.

3.7.5 Portability

- YOLO Robot system shall support any operating system such as Windows, Linux and MacOS in case of connection.
- Instructions and commands which are given to YOLO system shall be Python code.
- YOLO Robot system shall support extended Python libraries.

3.8 Supporting Information

YOLO Robot is an open-source hardware and software project. Users who have technical background on 3D printing, assembling hardware components and programming skills can assemble 3D-printed hardware components which are provided with STL documentation, along with the embedded computer Raspberry Pi and its software components. Moreover, researchers in psychology area can contribute to the project with experimenting children-robot interaction research. Also, users who have an interest in robot development can contribute to the project with providing new behavior codes to the system.