Software Architecture Description
YOLO Robot

Andaç Berkay Seval 2235521

Musa Furkan Zenbilci 2469203

Group No: 37

Table of Contents

List of Figures

List of Tables

Revision History

Version	Date
1.0	21.05.2022
1.1	05.06.2022

Table 1: Change History

1 Introduction

1.1 Purpose and objectives of the YOLO social robot

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 Stakeholders and their concerns

There are different group of users within the system.

- **Users:** 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 (Teachers, 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 create 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.
- **Developers:** Developers are the people whose main concern is to create software extensions to the YOLO Robot system such as new move types, LED light options and behavior types.

2 References

This document is prepared with respect to IEEE 42010-2011 standard:

42010-2011 – ISO/IEC/IEEE International Standard – Systems and software engineering – Architecture description

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

Rozanski N., Woods E. (2012). *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*.

Sommerville I., (2016). *Software Engineering*. Pearson Education Limited, Boston, USA

3 Glossary

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
OS	Operating System

Table 2: Glossary

4 Architectural Views

4.1 Context View

4.1.1 Stakeholder's use of this view

All stakeholders are involved in this view. Users, IT people and developers use this view to interact with the YOLO Robot system. Users use this view while playing with YOLO Robot. Developers use this view while creating software extensions to YOLO Robot software via Desktop API. IT People use this view while maintaining system's hardware and software and keeping the system open for software extensions.

4.1.2 Context Diagram

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.

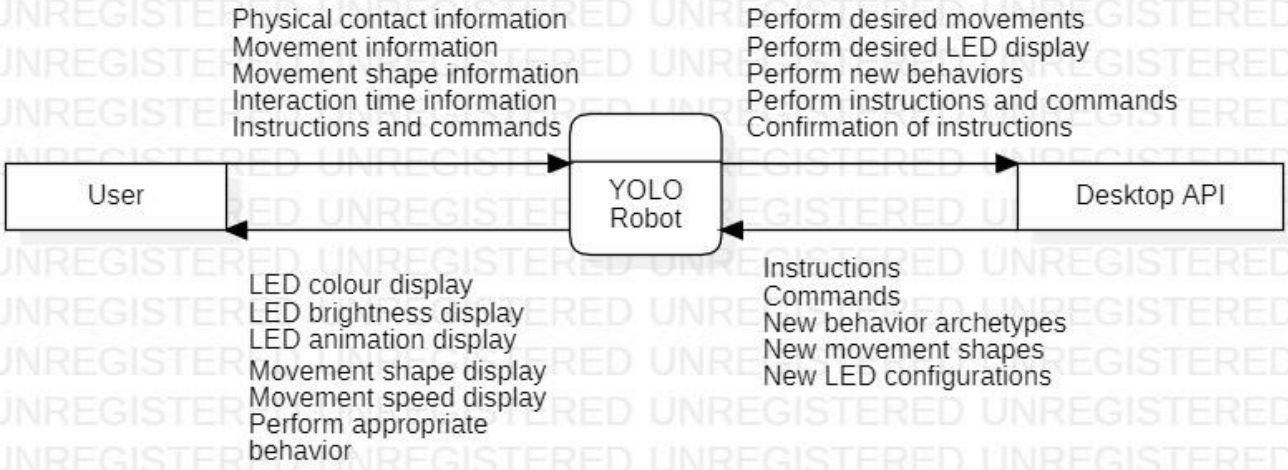

Figure 1: Context Diagram

4.1.3 External Interfaces

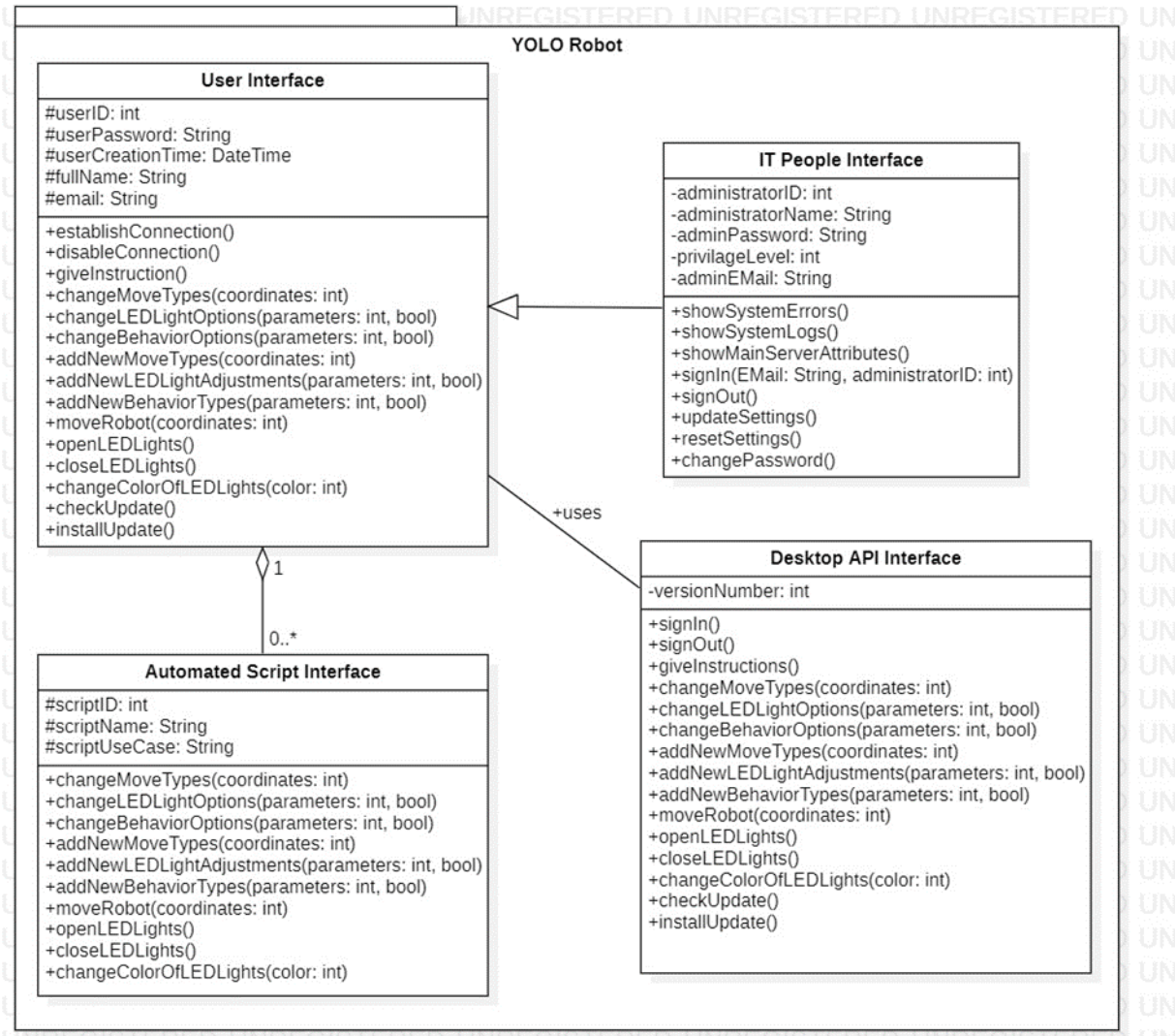

Figure 2: External Interfaces Class Diagram

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.

User Interface Operations:

- establishConnection(): connecting to YOLO Robot with a USB cable or wi-fi
- disableConnection(): disconnecting from YOLO Robot
- giveInstruction(): give instruction to YOLO Robot
- changeMoveTypes(coordinates: int): change current move type of YOLO Robot
- changeLEDLightOptions(parameters: int, bool): change current LED light options
- changeBehaviorOptions(parameters: int, bool): change current behavior options
- addNewMoveTypes(coordinates: int): add new move type to YOLO Robot
- addNewLEDLightAdjustments(parameters: int, bool): add new LED light adjustments
- addNewBehaviorTypes(parameters: int, bool): add new behavior type to YOLO Robot
- moveRobot(coordinates: int): move YOLO Robot in a desired way
- openLEDLights(): open LED lights of YOLO Robot
- closeLEDLights(): close LED lights of YOLO Robot
- changeColorOfLEDLights(color: int): change color of LED lights
- checkUpdate(): check for system updates
- installUpdate(): install system updates

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.

Automated Script Interface Operations:

- changeMoveTypes(coordinates: int): change current move type of YOLO Robot
- changeLEDLightOptions(parameters: int, bool): change current LED light options
- changeBehaviorOptions(parameters: int, bool): change current behavior options
- addNewMoveTypes(coordinates: int): add new move type to YOLO Robot
- addNewLEDLightAdjustments(parameters: int, bool): add new LED light adjustments
- addNewBehaviorTypes(parameters: int, bool): add new behavior type to YOLO Robot
- moveRobot(coordinates: int): move YOLO Robot in a desired way
- openLEDLights(): open LED lights of YOLO Robot
- closeLEDLights(): close LED lights of YOLO Robot
- changeColorOfLEDLights(color: int): change color of LED lights

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.

IT People Interface Operations:

- showSystemErrors(): show system failures
- showSystemLogs(): show past system logs
- showMainServerAttributes(): show YOLO Robot system server attributes
- signIn(Email: String, administratorID: int): sign into the system server
- signOut(): sign out from system server
- updateSettings(): update system server settings
- resetSettings(): reset system server settings
- changePassword(): change current password

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.

Desktop API Interface Operations:

- signIn(): sign into the system server
- signOut(): sign out from system server
- giveInstructions(): give instruction to YOLO Robot
- changeMoveTypes(coordinates: int): change current move type of YOLO Robot
- changeLEDLightOptions(parameters: int, bool): change current LED light options
- changeBehaviorOptions(parameters: int, bool): change current behavior options
- addNewMoveTypes(coordinates: int): add new move type to YOLO Robot
- addNewLEDLightAdjustments(parameters: int, bool): add new LED light adjustments
- addNewBehaviorTypes(parameters: int, bool): add new behavior type to YOLO Robot
- moveRobot(coordinates: int): move YOLO Robot in a desired way
- openLEDLights(): open LED lights of YOLO Robot
- closeLEDLights(): close LED lights of YOLO Robot
- changeColorOfLEDLights(color: int): change color of LED lights
- checkUpdate(): check for system updates
- installUpdate(): install system updates

4.1.4 Interaction Scenarios

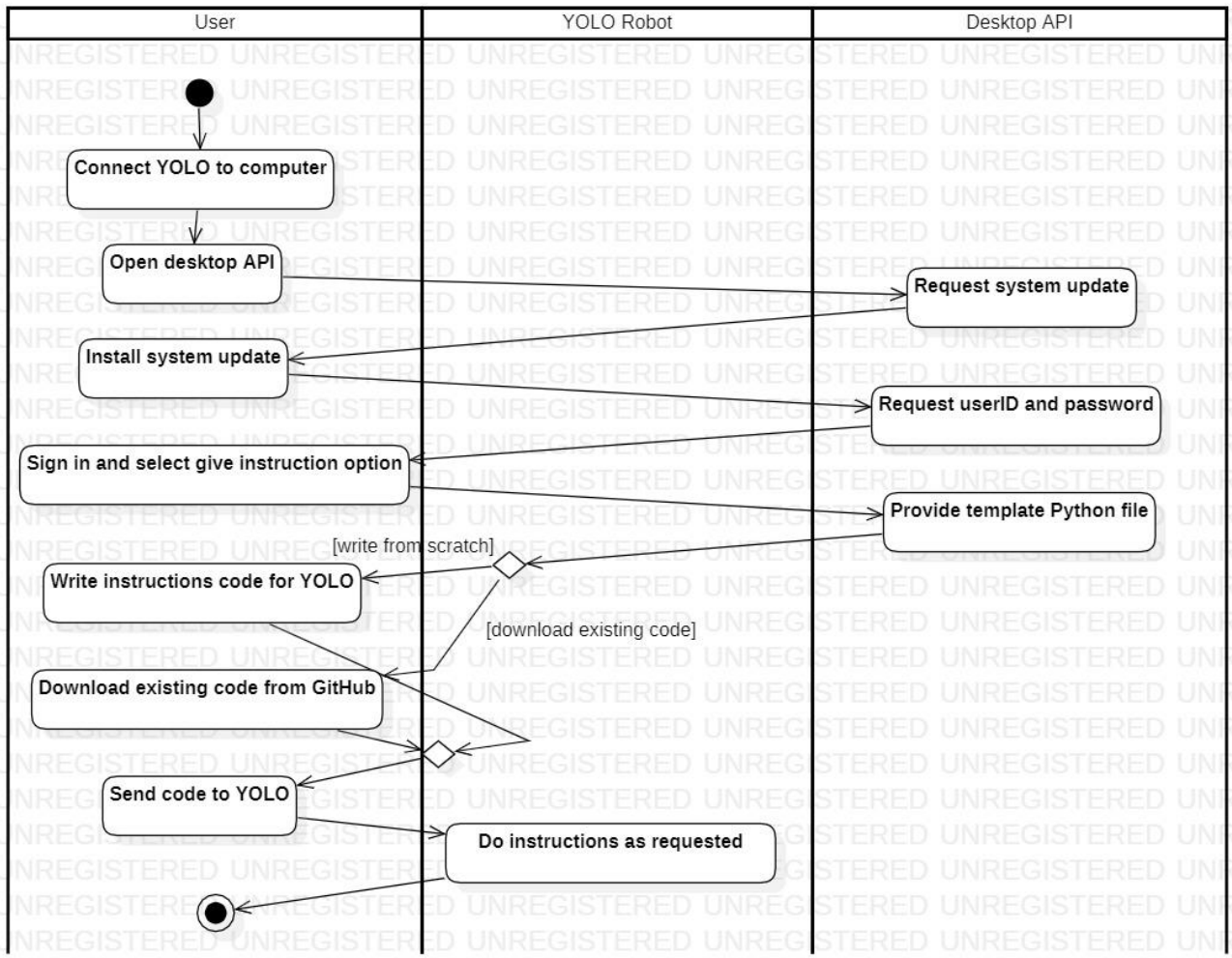

Figure 3: Activity Diagram of Giving Instructions to YOLO Robot

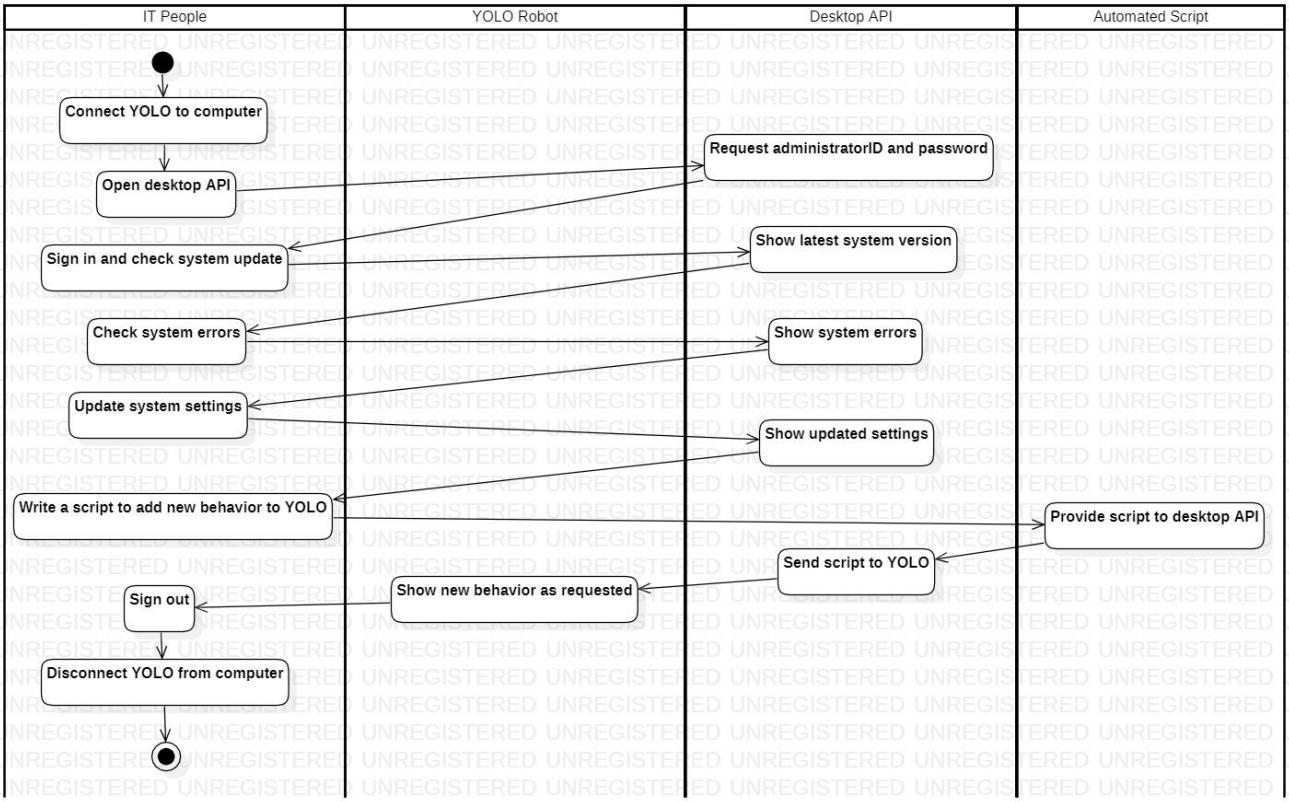

Figure 4: Activity Diagram of Updating System According to System Errors and Writing Automated Script for YOLO Robot

4.2 Functional View

4.2.1 Stakeholders' use of this view

All stakeholders are involved in this view. Users, IT people and developers use this view to decide functional capabilities, interfaces and interactions between the components of the YOLO Robot system. Users use this view while playing with YOLO Robot and provide new code and functionalities to YOLO Robot. Developers use this view while creating software extensions to YOLO Robot software via Desktop API. IT People use this view while maintaining system's hardware and software and keeping the system open for software extensions.

4.2.2 Component Diagram

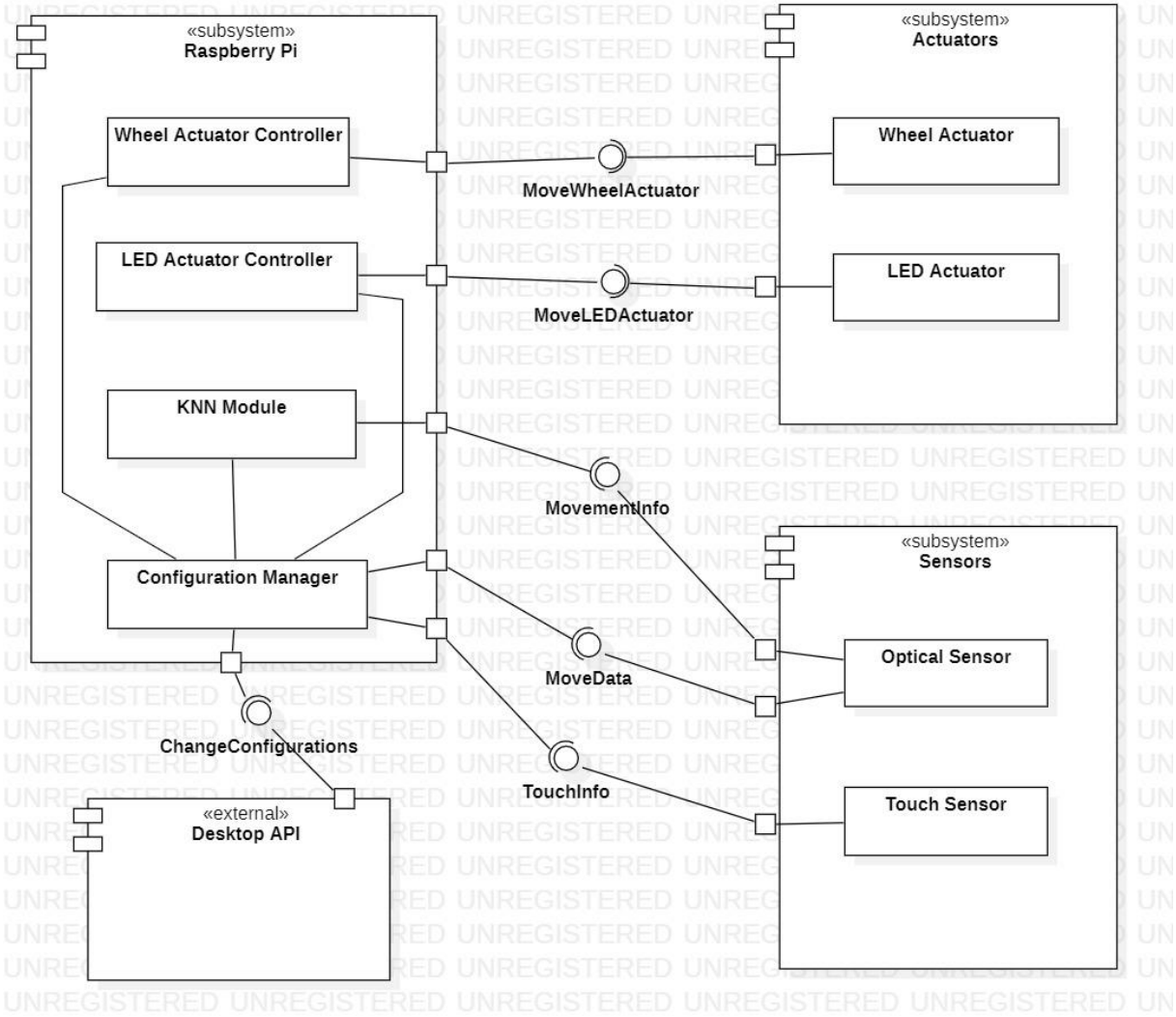

Figure 5: Component Diagram

- Raspberry Pi embedded computer is a subsystem of the YOLO Robot, and it controls main activities of the system. It controls wheel actuator and LED actuator according to given movement data and touch data. It has a configuration manager part which chooses which move type, LED light configuration and behavior option that YOLO Robot does in the moment. Also, it has a KNN module which obtains the shape from optical sensor movement information that user does with the YOLO Robot. There are relationships between configuration manager and KNN module, wheel actuator controller and LED actuator controller. KNN module sends shape information to configuration manager. Configuration manager sends move type and LED light configurations to actuator controller.
- Desktop API is an external component which provides user to change configurations of YOLO Robot. With desktop API, user can change move types, LED light configurations and behavior options of YOLO Robot via directly affecting configuration manager of Raspberry Pi unit.

- Actuators are subsystems of the YOLO Robot. There are two actuators: Wheel actuator and LED actuator. These actuators receive interfaces from Raspberry Pi unit according to given move data and LED light information so that they can move YOLO Robot properly and adjust LED lights for the situation.
- Sensors are subsystems of the YOLO Robot. There are two sensors: Optical sensor and Touch sensor. Optical sensor provides interface to both KNN module and configuration manager of the Raspberry Pi. KNN module predicts the shape with movement information and configuration manager configures move types according to that movement data. Touch sensor provides interface to configuration manager so that it can decide moving or not moving YOLO Robot according to touch feeling received from user.

4.2.3 Internal Interfaces

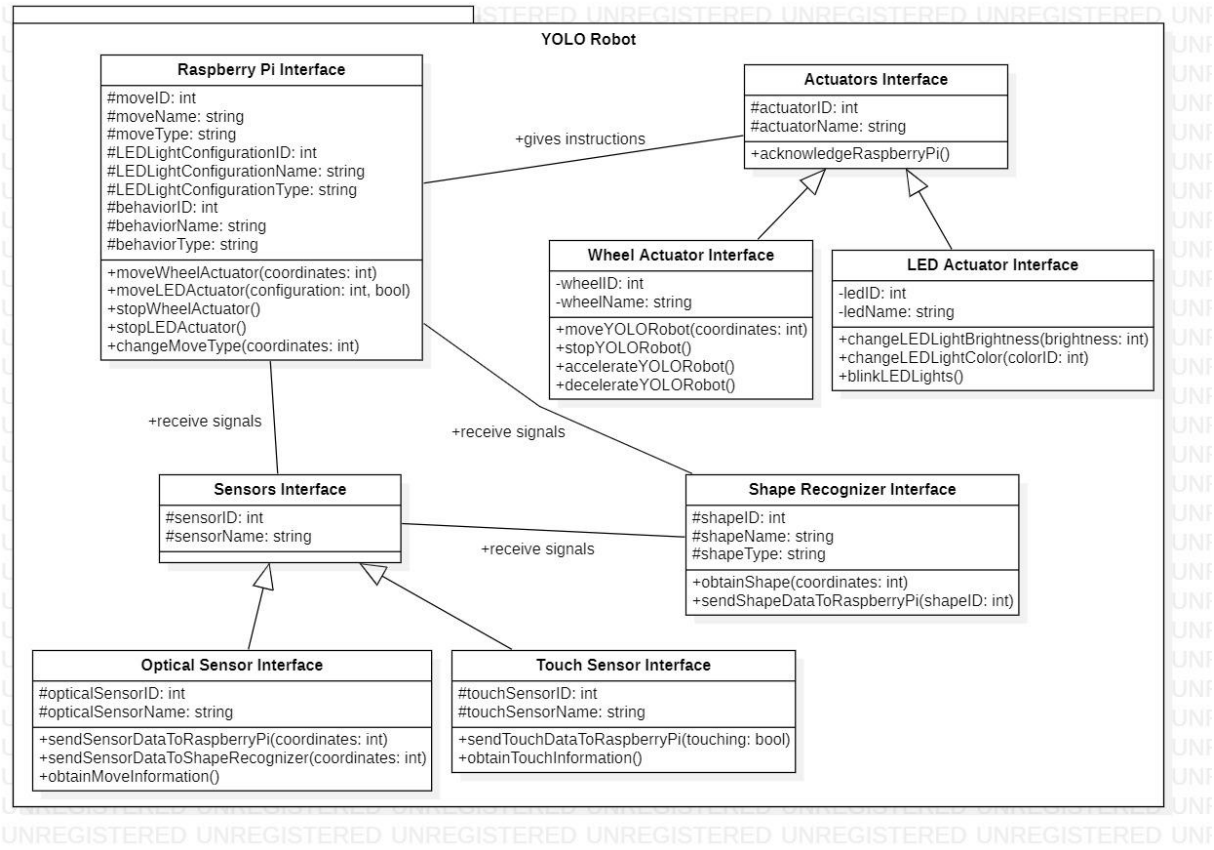

Figure 6: Internal Interfaces Class Diagram

Raspberry Pi Interface Operations:

- moveWheelActuator(coordinates: int): moves wheel actuator with respect to given coordinates
- moveLEDActuator(configuration: int, bool): moves LED actuator with respect to given configuration
- stopWheelActuator(): stops wheel actuator
- stopLEDActuator(): stops LED actuator
- changeMoveType(coordinates: int) : changes move type of wheel actuator

Sensors Interface Operations:

- sendSensorDataToRaspberryPi(coordinates: int): sends coordinate data received from optical sensor to Raspberry Pi
- sendSensorDataToShapeRecognizer(coordinates: int): sends coordinate data received from optical sensor to shape recognizer
- obtainMoveInformation(): obtains move information of YOLO Robot from optical sensor
- sendTouchDataToRaspberryPi(touching: bool): sends touching information to Raspberry Pi whether a user touches to YOLO Robot or not
- obtainTouchInformation(): obtains touch information of YOLO Robot from touch sensor

Actuators Interface Operations:

- acknowledgeRaspberryPi(): acknowledges Raspberry Pi with respect to actuator movement
- moveYOLORobot(coordinates: int): moves YOLO Robot with respect to given coordinates
- stopYOLORobot(): stops YOLO Robot
- accelerateYOLORobot(): accelerates YOLO Robot
- decelerateYOLORobot(): decelerates YOLO Robot
- changeLEDLightBrightness(brightness: int): changes LED Light brightness with respect to given brightness information
- changeLEDLightColor(colorID: int): changes LED Light color with respect to given colorID
- blinkLEDLights(): blinks LED Lights of YOLO Robot

Shape Recognizer Operations:

- obtainShape(coordinates: int): obtains movement shape of YOLO Robot
- sendShapeDataToRaspberryPi(shapeID: int): send shape data to Raspberry Pi

4.2.4 Interaction Patterns

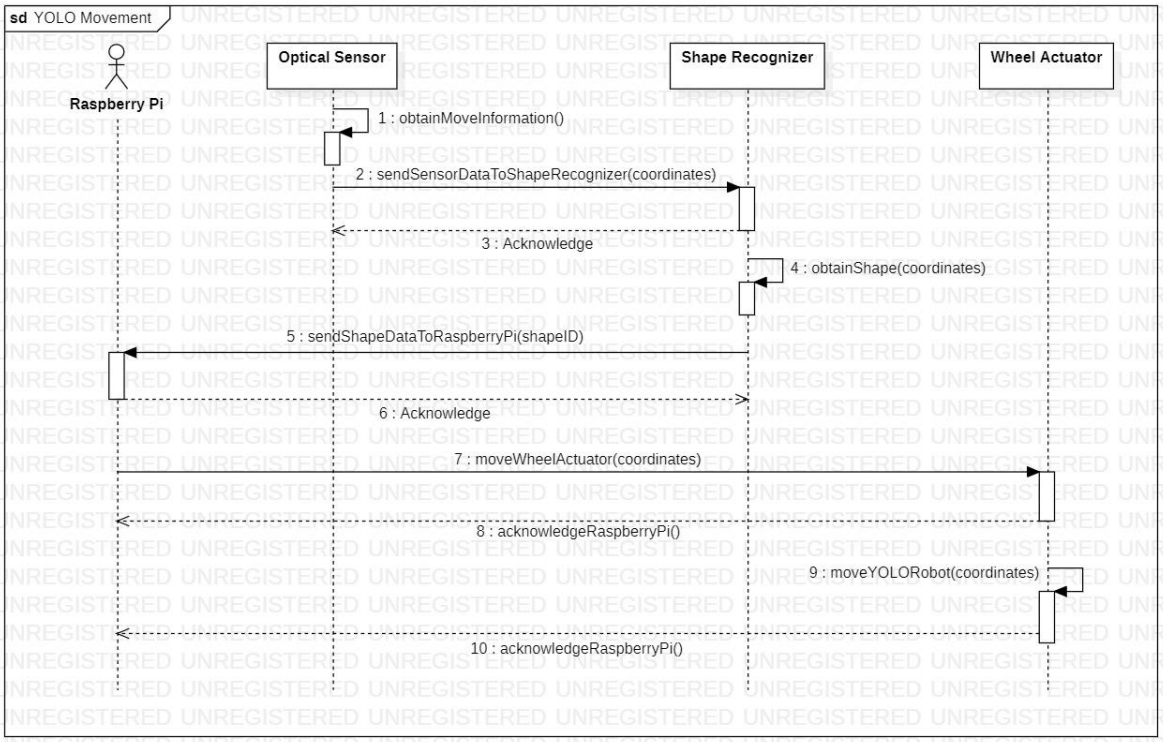

Figure 7: Sequence Diagram of YOLO Robot Movement

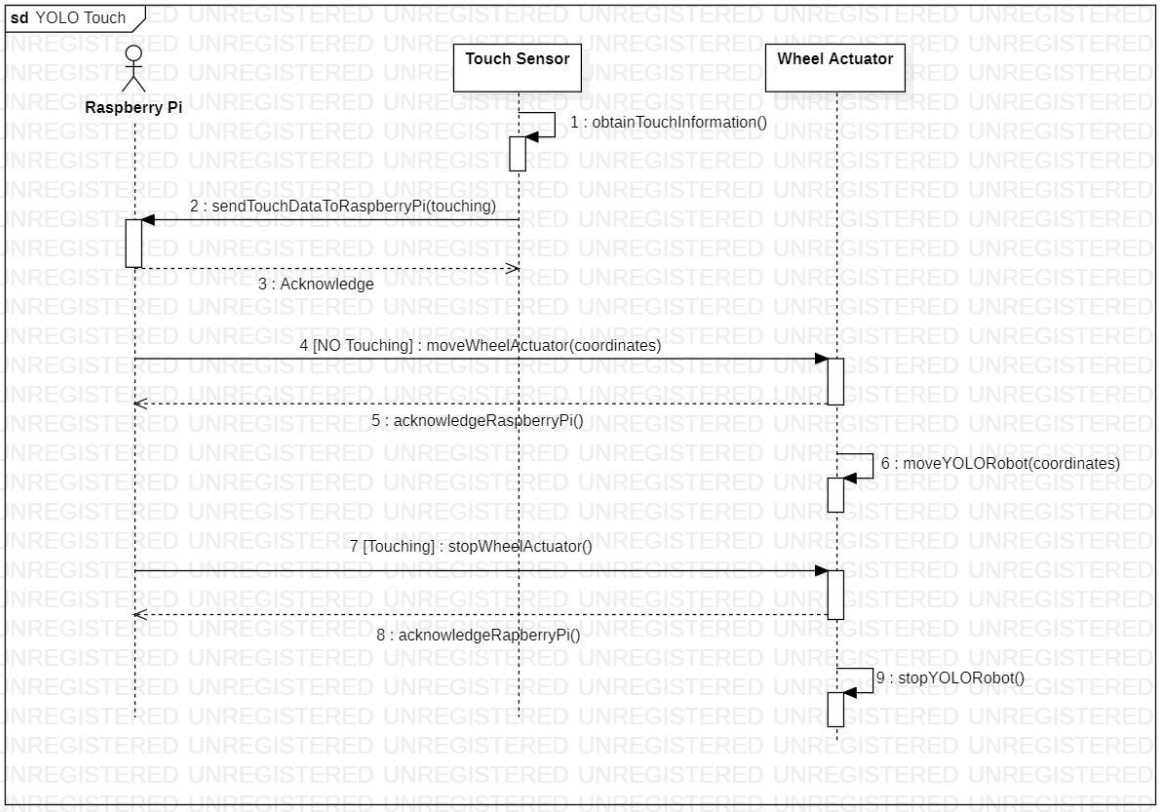

Figure 8: Sequence Diagram of Touching YOLO Robot

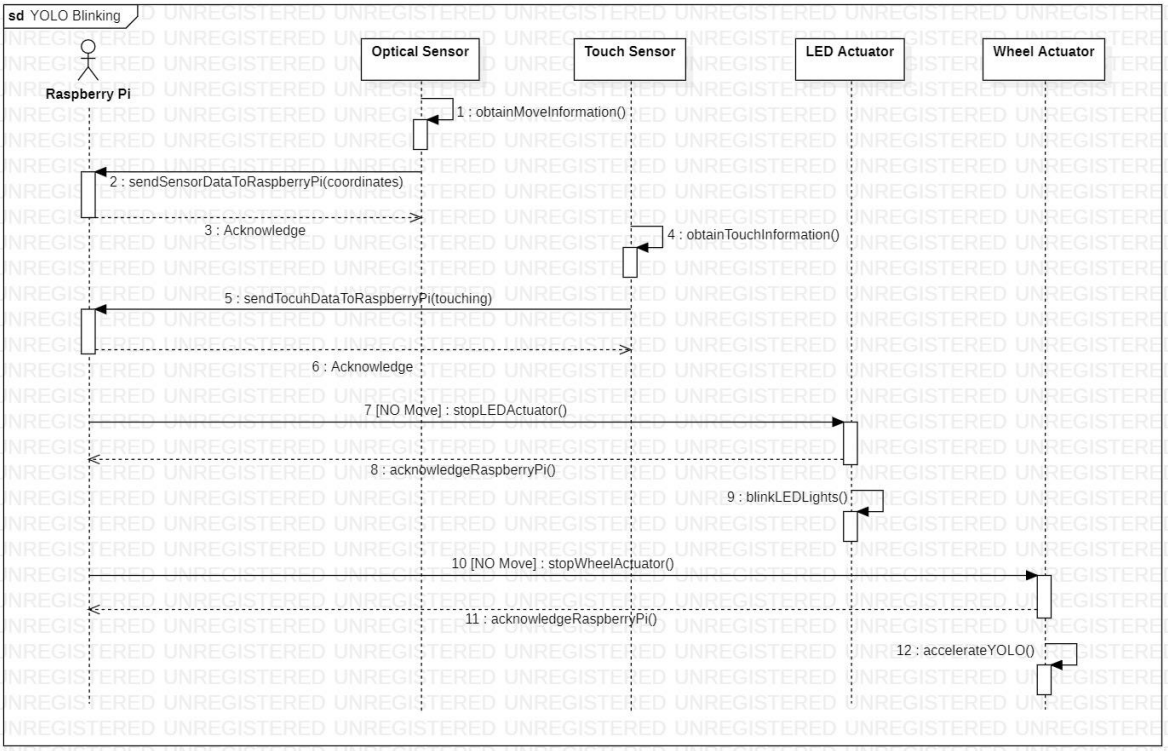

Figure 9: Sequence Diagram of Blinking YOLO

4.3 Information View

4.3.1 Stakeholders' use of this view

All stakeholders are involved in this view. Users, IT people and developers use this view to creating information flow in the system and information storage. Users use this view while changing and adding new move sets, LED light options, behavior configurations, giving instructions, checking and integrating hardware to the system and updating the system. Developers use this view while changing and adding new move sets, LED light options and behavior configurations. Administrators and maintainers use this view while observing system logs and system errors, changing instruction, hardware, behavior, LED light and move configurations, and updating/resetting system settings.

4.3.2 Database Class Diagram

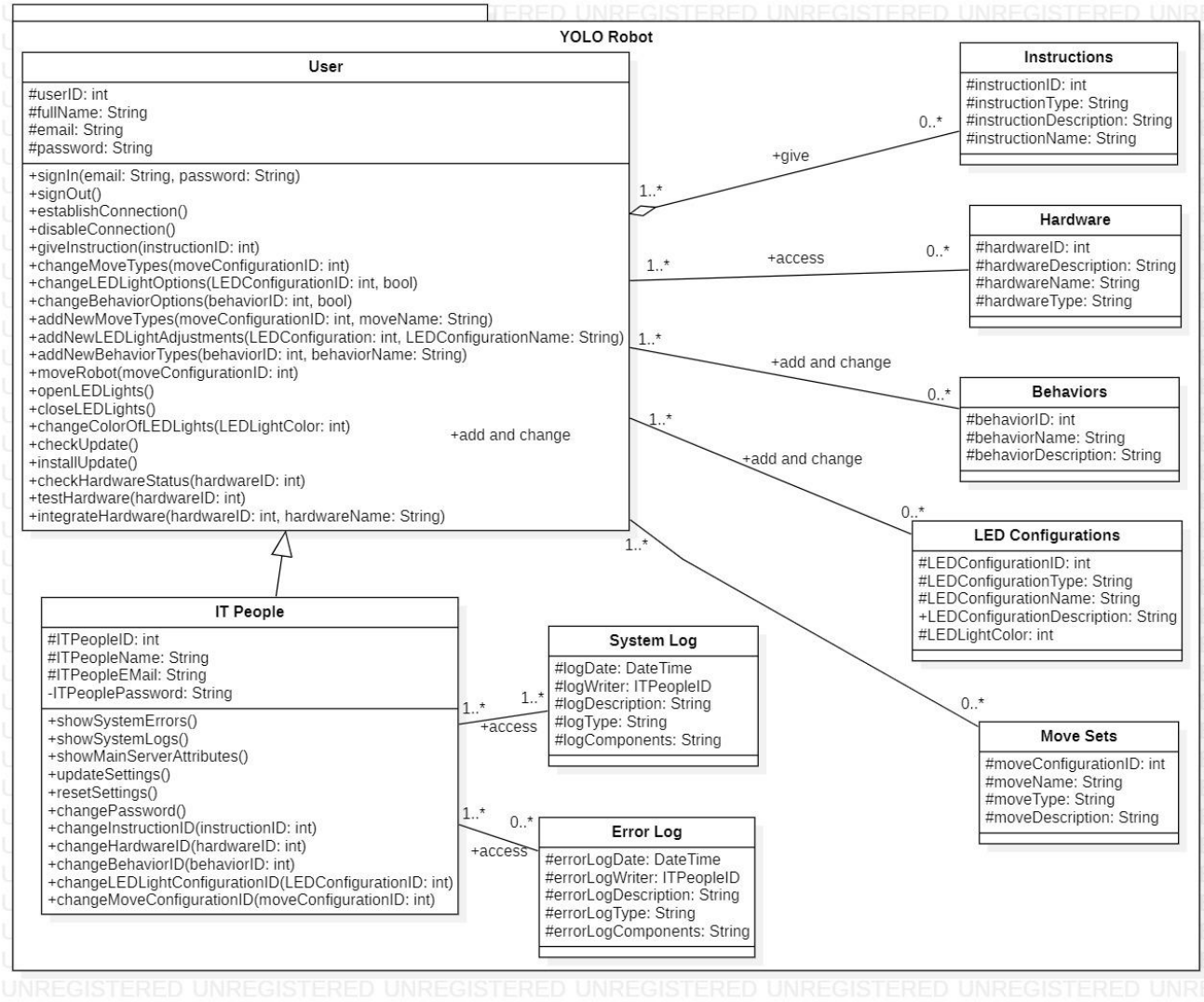

Figure 10: Database Class Diagram

- 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.

4.3.3 Operations on Data

Operation	**CRUD Operation**
signIn	Create: - Read: Customer Update: - Delete: -
signOut	Create: - Read: Customer Update: - Delete: -
establishConnection	Create: - Read: Customer Update: System Log Delete: -
disableConnection	Create: - Read: Customer Update: System Log Delete: -
giveInstruction	Create: - Read: Customer Update: System Log Delete: -
changeMoveTypes	Create: - Read: Customer Update: Move Configuration, System Log Delete: -
changeLEDLightOptions	Create: - Read: Customer Update: LED Light Configuration, System Log Delete: -
changeBehaviorOptions	Create: - Read: Customer Update: Behavior Configuration, System Log Delete: -
addNewMoveTypes	Create: Move Configuration Read: Customer

	Update: System Log Delete: -
addNewLEDLightAdjustments	Create: LED Light Configuration Read: Customer Update: System Log Delete: -
addNewBehaviorTypes	Create: Behavior Configuration Read: Customer Update: System Log Delete: -
moveRobot	Create: - Read: Customer Update: - Delete: -
openLEDLights	Create: - Read: Customer Update: - Delete: -
closeLEDLights	Create: - Read: Customer Update: - Delete: -
changeColorOfLEDLights	Create: - Read: Customer Update: - Delete: -
checkUpdate	Create: - Read: Customer Update: - Delete: -
installUpdate	Create: - Read: Customer Update: System Software, System Log Delete: Old System Software
checkHardwareStatus	Create: - Read: Customer Update: - Delete: -
testHardware	Create: - Read: Customer Update: System Log Delete: -
integrateHardware	Create: Hardware Configuration Read: Customer Update: System Log Delete: -
showSystemErrors	Create: - Read: IT People Update: - Delete: -

showSystemLogs	Create: - Read: IT People Update: - Delete: -
showMainServerAttributes	Create: - Read: IT People Update: - Delete: -
updateSettings	Create: System Settings Read: IT People Update: System Settings, System Log Delete: -
resetSettings	Create: - Read: IT People Update: System Log Delete: System Settings
changePassword	Create: - Read: IT People Update: Password, System Log Delete: -
changeInstructionID	Create: - Read: IT People Update: Instruction Configuration, System Log Delete: -
changeHardwareID	Create: - Read: IT People Update: Hardware Configuration, System Log Delete: -
changeBehaviorID	Create: - Read: IT People Update: Behavior Configuration, System Log Delete: -
changeLEDLightConfigurationID	Create: - Read: IT People Update: LED Light Configuration, System Log Delete: -
changeMoveConfigurationID	Create: - Read: IT People Update: Move Configuration, System Log Delete: -

Table 3: CRUD Operations

4.4 Deployment View

4.4.1 Stakeholders' use of this view

IT People and developers are involved in this view. IT people and developers use this view to decide runtime environment of the elements of the YOLO Robot system. Developers use this view while creating software extensions to YOLO Robot software via Desktop API and downloading third-party software to the system. IT People use this view while maintaining system's hardware and software, deciding hardware components of the system, deciding physical parts of the system and compatibility of the components.

4.4.2 Deployment Diagram

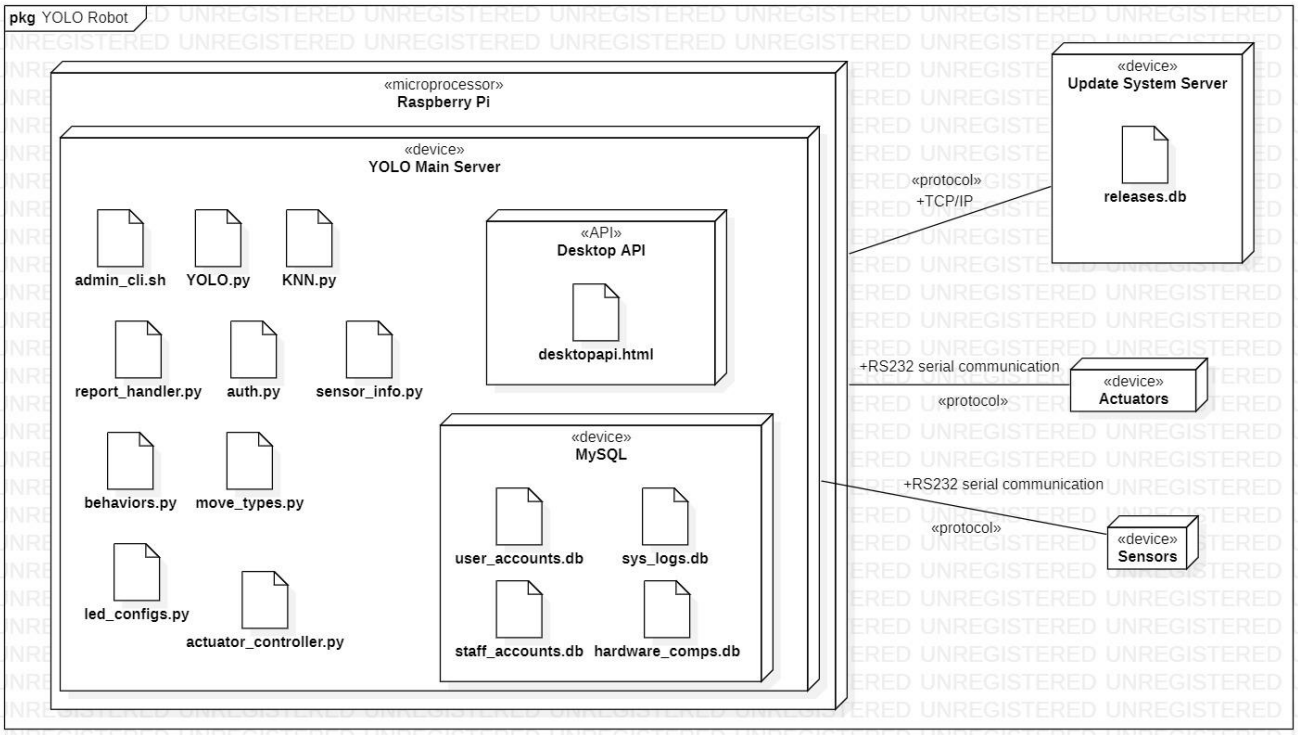

Figure 11: Deployment Diagram

- Python is the main language for executing programs in Raspberry Pi.
- MySQL is used as data management system to store system logs, user accounts, staff accounts and hardware component features.
- In the main server of YOLO, there are predefined behavior, move type and led configuration files.
- All sensors and actuator controller information are stored in the main server.
- There is a KNN module in the main server.
- Desktop API is provided from main server of YOLO to users in case of connection to a computer.
- There is an update server for the main server, and it is connected to the main server through a TCP/IP protocol.
- Wheel actuator, LED actuator, optical sensor and touch sensor are connected to Raspberry Pi through RS232 serial communication protocol.

4.5 Design Rationale

Design Rationale for Context View:

- In context view, main purpose is to decide context and boundary of the system. Thus, main priority is finding external entities that are communicate with YOLO.
- In context diagram, all kinds of stakeholders (users, developers, parent, teachers, researchers, IT People) are involved in the model as users.
- Although desktop API is provided from Raspberry Pi, one could think it as a graphical interface through a computer.
- With a connection to a computer, one could write code from scratch or download preexisting code from GitHub and send to the YOLO Robot.
- Desktop API provide users to give instant instructions, changing behavior options, move types and LED configurations or adding new behaviors, move types and LED configurations to YOLO Robot.
- 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.
- IT People are administrators and maintainers of the system.
- IT People have unique IDs and passwords which are saved in the system database.
- IT People can see system errors, system logs.
- IT People can update system settings. Also, they can reset system settings.

Design Rationale for Functional View:

- Users, IT people and developers use this view to decide functional capabilities, interfaces and interactions between the components of the YOLO Robot system.
- Users use this view while playing with YOLO Robot and provide new code and functionalities to YOLO Robot.
- Developers use this view while creating software extensions to YOLO Robot software via Desktop API.
- IT People use this view while maintaining system's hardware and software and keeping the system open for software extensions.
- Raspberry Pi embedded computer is a subsystem of the YOLO Robot, and it controls main activities of the system. It controls wheel actuator and LED actuator according to given movement data and touch data. It has a configuration manager part which chooses which move type, LED light configuration and behavior option that YOLO Robot does in the moment.
- Raspberry Pi has a KNN module which obtains the shape from optical sensor movement information that user does with the YOLO Robot. There are relationships between configuration manager and KNN module, wheel actuator controller and LED actuator controller. KNN module sends shape information to configuration manager. Configuration manager sends move type and LED light configurations to actuator controller.
- Actuators are subsystems of the YOLO Robot. There are two actuators: Wheel actuator and LED actuator. These actuators receive interfaces from Raspberry Pi unit according to given move data and LED light information.

- Sensors are subsystems of the YOLO Robot. There are two sensors: Optical sensor and Touch sensor. Optical sensor provides interface to both KNN module and configuration manager of the Raspberry Pi. Touch sensor provides interface to configuration manager so that it can decide moving or not moving YOLO Robot according to touch feeling received from user.

Design Rationale for Information View:

- Users, IT people and developers use this view to creating information flow in the system and information storage.
- Users use this view while changing and adding new move sets, LED light options, behavior configurations, giving instructions, checking and integrating hardware to the system and updating the system.
- Developers use this view while changing and adding new move sets, LED light options and behavior configurations.
- Administrators and maintainers use this view while observing system logs and system errors, changing instruction, hardware, behavior, LED light and move configurations, and updating/resetting system settings.
- 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.
- 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 LED configurations of the system.
- Each LED configuration has a unique ID and name.
- Users can add and change move types of the system.
- Each move type has a unique ID and name.

Design Rationale for Deployment View:

- IT people and developers use this view to decide runtime environment of the elements of the YOLO Robot system.
- Developers use this view while creating software extensions to YOLO Robot software via Desktop API and downloading third-party software to the system.
- IT People use this view while maintaining system's hardware and software, deciding hardware components of the system, deciding physical parts of the system and compatibility of the components.
- Python is the main language for executing programs in Raspberry Pi.
- MySQL is used as data management system to store system logs, user accounts, staff accounts and hardware component features.

- All sensors and actuator controller information are stored in the main server.
- There is a KNN module in the main server.
- Desktop API is provided from main server of YOLO to users in case of connection to a computer.
- In the main server of YOLO, there are predefined behavior, move type and led configuration files.
- Wheel actuator, LED actuator, optical sensor and touch sensor are connected to Raspberry Pi through RS232 serial communication protocol.
- There is an update server for the main server, and it is connected to the main server through a TCP/IP protocol.