FROBOT – TEAM08

CPSC 362

Abstract

This robot is designed to be used in a hospital system to limit the exposure to COVID-19

Instructor: Chang-Hyun Jo

Submission Date: 6/10/2020

Garinn Morton, Naoki Atkins

gmorton2@csu.fullerton.edu, naokishami@csu.fullerton.edu

Revision History

Element	Who	Description
Functional Requirement	Garinn Morton	Added FR01-FR06
Functional Requirement	Naoki Atkins	Added FR07-FR10
Nonfunctional Requirement	Garinn Morton	Added NFR01-NFR05
UC Text	Garinn Morton	Added UC01-UC05
UC Text	Naoki Atkins	Added UC06-UC10
UC Diagram	Garinn Morton	Added UCD
FR to Use Case	Garinn Morton	Added FR to Use Case
Swimlane Diagram	Garinn Morton	Added SLD
Activity Diagram	Garinn Morton	Added AD
Class Diagram	Naoki Atkins	Added Class Diagram
Class Responsibility Diagram	Naoki Atkins	Added CRC
State Diagram	Naoki Atkins	Added State Diagram
Sequence Diagram	Garinn Morton	Added Sequence Diagram
State Diagram	Garinn Morton	Added State Diagram

Table of Contents

Communication

This robot was designed to be used in a hospital system. It drastically decreases the number of times a medical professional will have to come in direct contact with a sick patient. The primary usage of this robot will be to: register new patients, diagnose patients, disinfect rooms, and deliver medical supplies.

This robot will be charging at the entrance of the hospital, waiting for a command.

When it sees a new person walk in to the hospital, it will enter the register state. In the register state, the robot will ask the person if they are a visitor or if they are sick. If the person is a visitor, the robot will ask who they are seeing and give them the room number. If the person is sick, the robot will proceed to ask the patient for their name and phone number and assign them a room number.

Upon request from a nurse, the robot will diagnose a patient given some details. The robot will ask the nurse if they would like the robot to diagnose just one patient, or a range of patients, or all patients. The robot will then diagnose these patients according to the nurse's response.

If a nurse walks up to the robot and ask it to disinfect a room, it will enter the disinfect state. It will first clarify with the nurse whether they want one room or a range of rooms or if they want all the rooms to be disinfected. The robot will then proceed to disinfect the rooms accordingly.

The robot will also deliver medical supplies to patients. The nurse will need to walk up to the robot at the charging station and state which patients they would like the medical supplies delivered to.

Through the use of technology, we will be able to mitigate the spread of the coronavirus and help ensure a safer tomorrow.

Planning

Estimating the project

- Budget: $10,000
- People: 2 Software Engineers
- Resources: Lucidchart, draw.io

Scheduling

- Time: 5 weeks
- Work: 80 hours/week

Planning

Project scope: To create a robot that will perform all the functional requirements.

LOC: 50,000

Work Task	Planned Start	Actual start	Planned complete	Actual complete	Assigned person	Effort Allocated	Notes
FR	6/1	6/1	6/2	6/2	GM	2 p-d	
NFR	6/1	6/1	6/2	6/2	NA	1 p-d	
UC text	6/2	6/2	6/3	6/3	GM	3 p-d	
UCD	6/3	6/3	6/4	6/4	GM	2 p-d	
ACD	6/4	6/3	6/5	6/4	GM	3 p-d	
CRC	6/4	6/4	6/4	6/4	NA	4 p-d	
AD	6/4	6/4	6/4	6/4	GM	2 p-d	
SLD	6/5	6/6	6/6	6/7	GM	2 p-d	
SD	6/5	6/5	6/5	6/6	NA	1 p-d	
AD	6/18	6/19	6/20	6/20	NA	2 p-d	
CLD	6/18	6/18	6/21	6/20	NA	1 p-d	
UE/ID	6/20	6/20	6/23	6/23	GM	3 p-d	
MAD	6/19	6/21	6/20	6/21	GM	2 p-d	
PBD	6/19	6/20	6/21	6/22	GM	3 p-d	
C:C	6/22	6/24	6/24	6/24	NA	4 p-d	
C:T	6/22	6/22	6/23	6/24	GM	2 p-d	
CT	6/23	6/24	6/25	6/25	GM	2 p-d	

Waterfall Process

Communication (done) -> Planning (done) -> Modeling Requirements Analysis (done) -> Modeling Design -> Construction -> Deployment

Modeling: Requirements Analysis

Functional Requirements

Requirement ID	Requirement Statement	Must/Want	Comments
FR01	The robot shall be able to speak a few words and a few statements	Must	Less than 10 words/statements
FR02	The robot shall show its text/script/sign-language on the screen	Must	
FR03	The robot shall listen to your order	Must	By using either voice recognition or programmable buttons
FR04	The robot shall be able to move around the designated places	Must	By using censors, camera to detect surroundings
FR05	The robot shall be able to hold objects by their hands	Must	Object must be less than 10 pounds
FR06	The robot shall be able to release whatever object it is holding	Must	
FR07	The robot will take people's temperatures using a thermal-imaging camera	Want	Person can be standing close to the robot or at the most, 100 meters away
FR08	Kill virus with ultraviolet light	Want	
FR09	Give a coronavirus diagnosis	Want	Using speech, text, and email
FR10	Deliver masks, sanitizer, and any medications for those in need	Want	Given a person that the robot identifies is potentially in need of medical supplies, the robot will roll over to them, and ask if they would like to receive anything

Non-Functional Requirements

Requirement ID	Requirement Statement	Must/Want	Comments
NFR01	Give a temperature reading that's accurate to 0.1 degree Fahrenheit	Must	

NFR02	Give a diagnosis that is correct 99.9% of the time	Want	Must evaluate
NFR03	Deliver 10 items a minute	Want	Delivery can be done in a shopping mall
NFR04	The sellers/manufacturers will provide/install a set of default/example applications when the customers purchase the robots	Must	The customer can run the robot 'as pre-programmed', or they can change the robot program only if they want.
NFR05	The robots lifetime will be tested to devise a warranty plan in case the robot malfunctions in anyway	Want	

Use Cases Text

UC01: Speak

- Primary Actor: User
- Goal in context: The robot shall be able to speak a few words and a few statements when commanded.
- Preconditions: The robot must have a functioning speaker.
- Trigger: User must press the speech button on the robot's remote controller.
- Scenario:
    - User presses the on button on the remote controller.
    - User presses the speak button on the remote controller.
    - If it is the first time the button has been pressed after it has been turned on then the robot shall say a greeting message.
    - User presses the speak button on the remote controller for the second time.
    - Robot says a randomly generated question or statement that is not a greeting message.
- Exceptions:
    - The speaker has been damaged and the robot is not able to say any statements.
    - May only speak 10 words or fewer at a time.

UC02: Display

- Primary Actor: User
- Goal in context: The robot shall show the words spoken by it in text/script/sign-language on a display screen.
- Preconditions: The robot must have a functioning LED screen to display text.
- Trigger: User must press the speech button on the robot's remote controller.
- Scenario:

    - User presses the on button on the remote controller.
    - User presses the speech button on the remote controller.
    - Whatever the robot says is then displayed as text on an LED screen
- Exceptions:
    - If the screen is malfunctioning then the text will not be displayed.

UC03: Listen

- Primary Actor: User
- Goal in context: The robot shall listen to your order and preform said order.
- Preconditions: The robot must have a functioning microphone to pick up the user's command.
- Trigger: User must start command with "Hey robot….".
- Scenario:
    - User presses the on button on the remote controller.
    - User says "Hey robot, what is the weather like to today?".
    - The robot then replies with the degrees in Fahrenheit, the humidity, and the wind speed of the city you are in.
- Exceptions:
    - Command does not start with "Hey robot".
    - The robot does not understand what the command the user gave to it.
    - The robot is unable to perform the command the user gave to it.

UC04: Move

- Primary Actor: User
- Goal in context: The robot shall keep taking 4 steps forward until an obstacle is measured to be 1 foot in front of the robot, then a 90-degree rotation shall be performed to move around said obstacle.
- Preconditions: The robot must have a functioning front camera that sees 180 degree of the surrounding area.
- Trigger: User must press the locomotion button on the robot's remote controller.
- Scenario:
    - User presses the on button on the remote controller.
    - User presses the locomotion button on the remote controller.
    - The robot scans the area and takes 4 steps forward while measuring if any obstructions are within 1 foot of it.
    - Once an obstacle has been found to be within range the robot stops and rotates 90 degrees and begins to scan again.
- Exceptions:
    - Obstacles are within 1 foot of the robot on all 360 degrees of its surroundings.
    - Obstacle is not big enough to be picked up by camera.

UC05: Hold

- Primary Actor: User
- Goal in context: The robot shall be able to hold objects.
- Preconditions: The object must be a weight of 10 pounds or less
- Trigger: User must press the hold button on the robot's remote controller.
- Scenario:
    - User presses the on button on the remote controller.
    - User puts an object within the grasp radius of the robot
    - User presses the hold button on the remote controller.
    - The robot clamps down on whatever the user wants it to hold
- Exceptions:
    - The object is greater than 10 pounds

UC06: Release

- Primary Actor: User
- Goal in context: The robot shall release the object that it is holding
- Preconditions: Robot shall be able to receive voice commands
- Trigger: User must press the release button or command by voice
- Scenario:
    - User: press 'Release' button and/or issues a command by speech
    - Robot: releases the object that it is holding by setting it down on a surface
- Exceptions:
    - Robot is currently not holding onto any object

UC07: Take Temperature

- Primary Actor: User
- Goal in context: The robot shall use its thermal imaging camera to take the temperature of all observable people in its view. The robot must be able to show the view from its thermal imaging camera on its display. In this thermal display, people will have a temperature reading written above their heads so that the user can easily see who has what temperature
- Preconditions: Display that shows input from the thermal imaging camera
- Trigger: User presses 'Take Temperature' button or issues a voice command
- Scenario:
    - User: press 'Take Temperature' button and/or issues a command by speech
    - Robot: turns on its thermal imaging camera
    - Robot: labels each person with their temperature shown above their head
    - This continues until the user has pressed the 'Stop' button
- Exceptions:
    - Person is too far away

UC08: Sanitize

- Primary Actor: User
- Goal in context: The robot will use its UV light to disinfect a room
- Preconditions: The robot must be able to move around objects
- Trigger: User must press the sanitize button
- Scenario:
    - User: presses ‘Sanitize’ button and/or issues a command by speech
    - Robot: spins its UV light in all directions ensuring that there is no surface that is left untreated
    - Robot: moves to a new location to treat another area
    - This continues for 15 min and the robot returns back to the user
- Exceptions:
    - There’s a surface that’s too high for the robot to reach.
        - For example, the top of a bookshelf

UC09: Test

- Primary Actor: User
- Goal in context: The robot shall test a person for COVID-19
- Preconditions: The robot must be able to receive input from a person and send text messages
- Trigger: User must press ‘Test for COVID-19’ button
- Scenario:
    - User: press ‘Test for COVID-19’ button and/or issues a command by speech
    - Robot: will ask the person for their name and phone number
    - User: will enter their information
    - Robot: will proceed to take the person’s temperature
    - Robot: robot will also perform a saliva test
    - Robot: will notify the person that the test is in process and that they will be contacted later
    - Robot: will send a text message to the person who has been tested about their result
- Exceptions:
    - None

UC10: Deliver

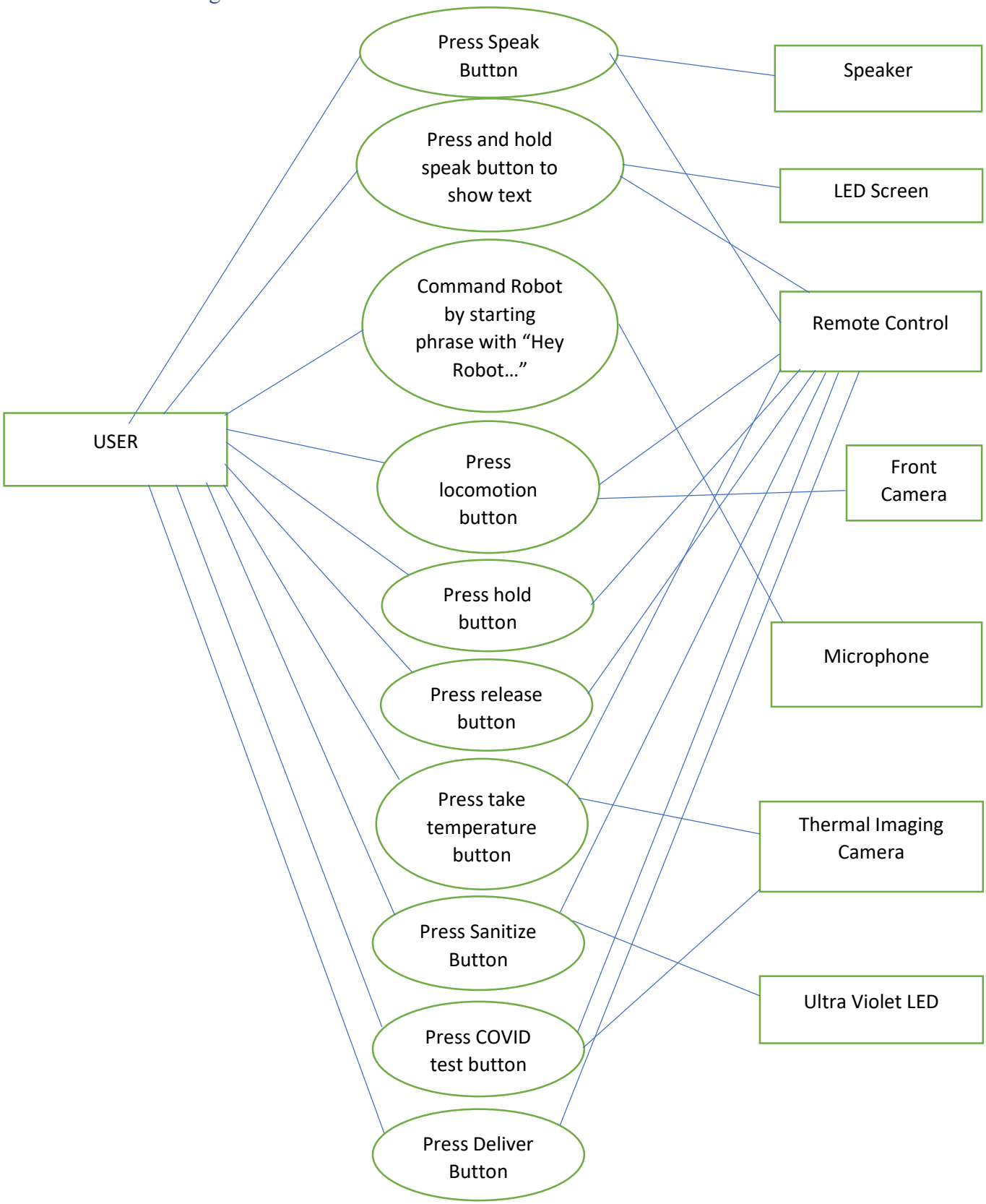
- Primary Actor: User
- Goal in context: The robot shall deliver masks, sanitizer, and/or any other medical items to those in need.
- Preconditions: The robot must be able to move and have a storage of medical items. The robot must be able to take voice commands and respond to pressed buttons
- Trigger: User must press the deliver button
- Scenario:
    - User: presses ‘Deliver’ button and/or issues a command by speech

    - Robot: will first scan its area for people who are not wearing masks
    - Robot: If there are people who are not wearing masks, the robot will approach them and ask if they would like to have a mask and/or any other medical item
    - Robot: If there is no person without a mask, the robot will use its thermal imaging camera to detect those who have a temperature of 99.5 deg Fahrenheit or higher
    - Robot: The robot will then proceed to approach these people and ask if they would like to receive any medical items that the robot currently carries
    - Robot: If there is no person with a fever, the robot will stay still in its current position and display a text on its screen reading "Would you like any medical items?"
- Exceptions:
    - The robot runs out of medical items

FR - Use Case Correspondence

Requirement ID	Requirement Description	Corresponding Use Case
FR01	The robot shall be able to speak a few words and a few statements	UC1- Press speech button to have robot say preprogrammed statements
FR02	The robot shall show its text/script/sign-language on the screen	UC2 - Press and hold speak button to show text
FR03	The robot shall listen to your order	UC3 - Command Robot by starting phrase with "Hey Robot…"
FR04	The robot shall be able to move around the designated places	UC4 - Press locomotion button
FR05	The robot shall be able to hold objects by their hands	UC5 - Press hold button to grab objects
FR06	The robot shall be able to release what ever object it is holding	UC6 - Press release button to release objects
FR07	The robot will take people's temperatures using a thermometer or with a thermal-imaging camera	UC7 - Press temperature button to use thermal imagining camera to take temperature of user
FR08	Kill virus with ultraviolet light	UC8 - Press Sanitize button to use UV light to kill virus
FR09	Tell a person if they have the coronavirus or not	UC9 - Press COVID Test button to test user for COVID-19
FR10	Deliver masks, sanitizer, and any medications for those in need	UC10 - Press deliver button to deliver medical needs to users

Use Case Diagram



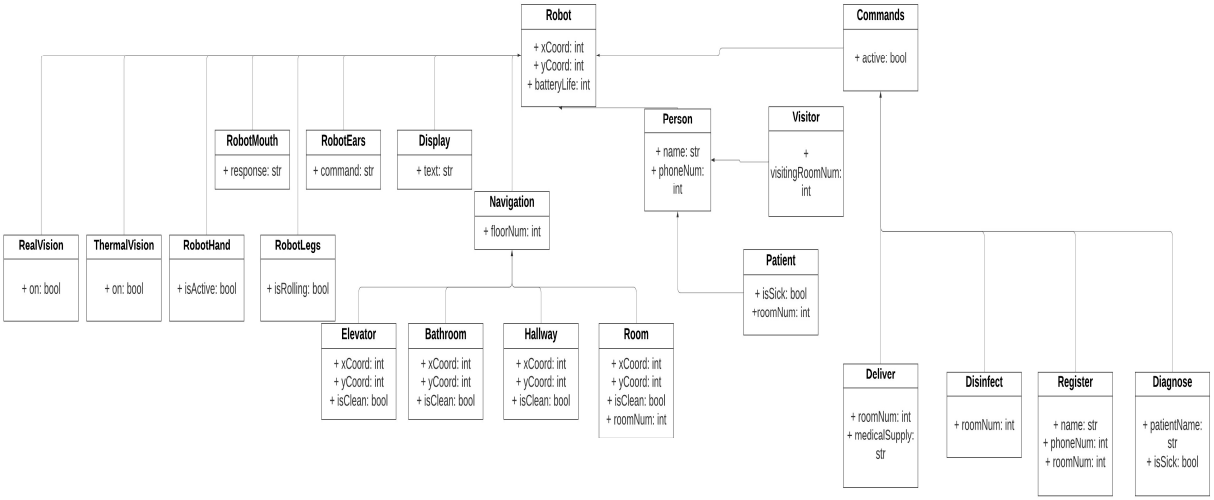
Class Diagram

- The class diagram was constructed by looking at all the use cases and performing a grammatical parse.

- After the grammatical parsing was done, all the nouns were taken out and evaluated as potential classes

- Potential classes were then evaluated further to identify all the final classes

Potential Class	General Classification
words	thing
screen	external entity
order/command	external entity
place	external entity
object	thing
room	external entity
nurse	role
visitor	role
patient	role
thermal imaging camera	external entity
coordinates	thing
is clean	thing



Class Responsibility Collaborator

CRC cards

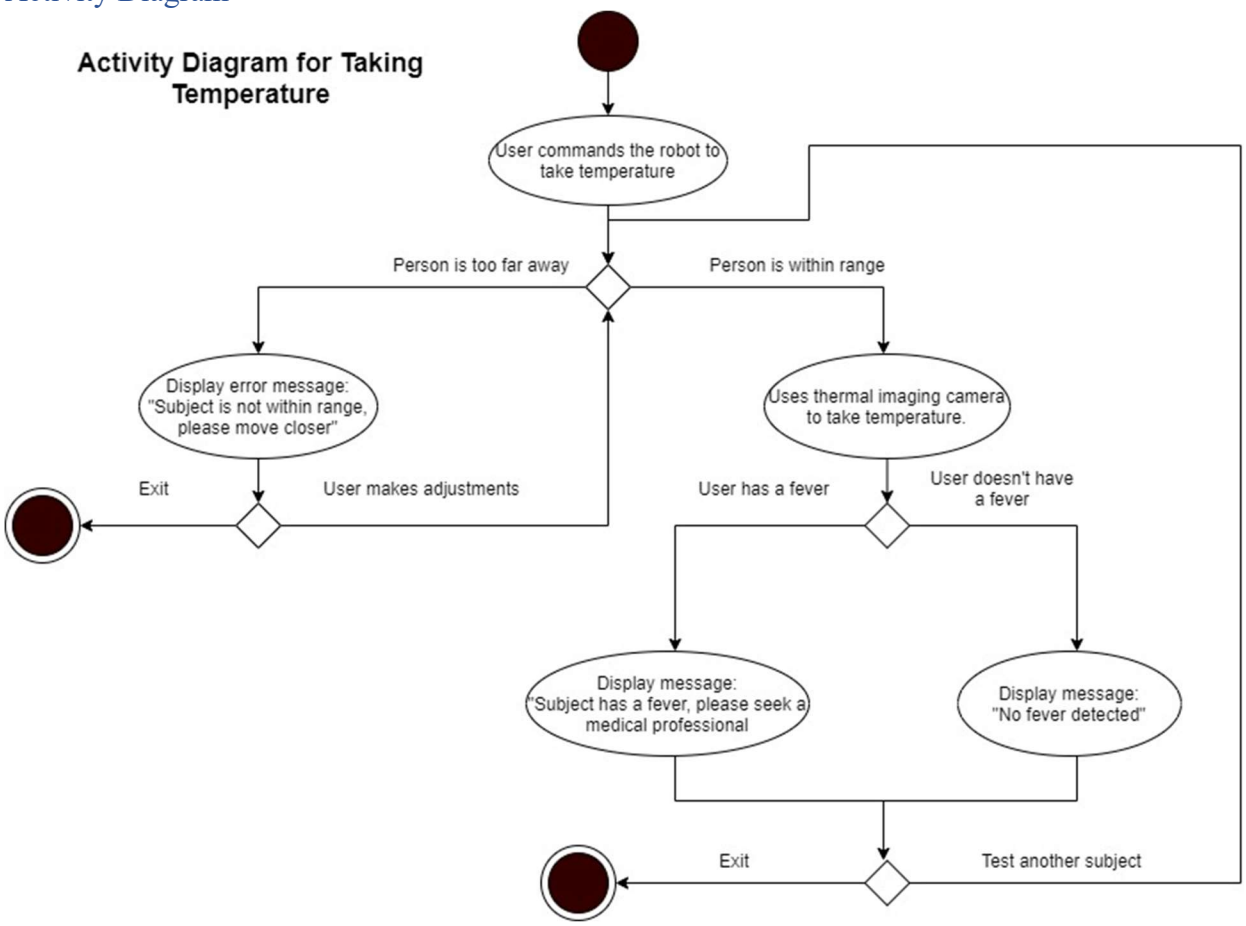
Naoki Atkins | June 10, 2020

Room	
- allocate a room to a Person	- Person
- let the robot know if the room is disinfected or not	- Disinfect
- coordinates	

Movement	
- detect objects in the way	- Camera
- move to a certain location	- Navigation
- go back to the charging location	

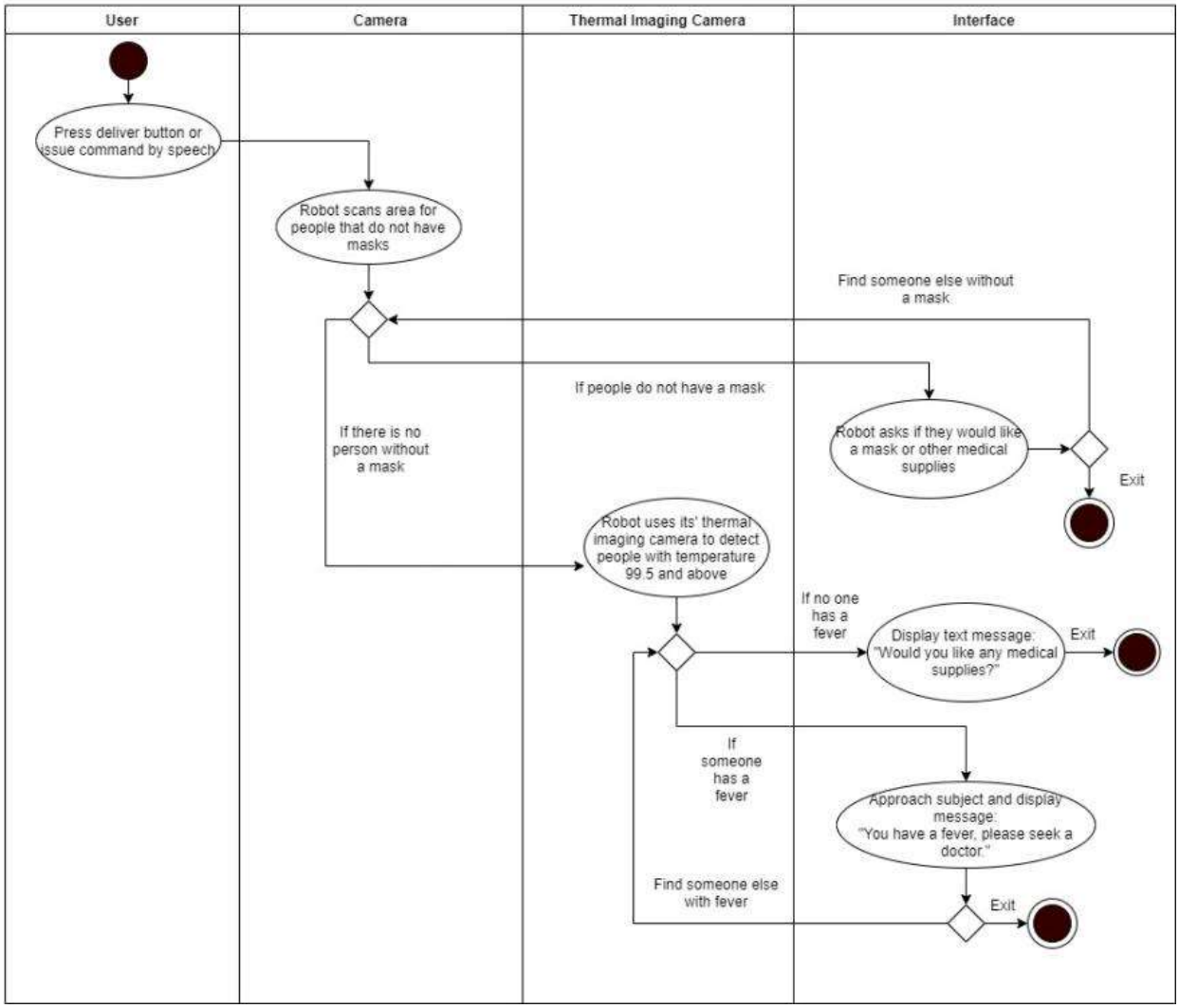
Register	
- register a Patient	- Patient
- assign a Patient an empty room	- Room
- tell a Visitor which room to visit	- Room

Activity Diagram

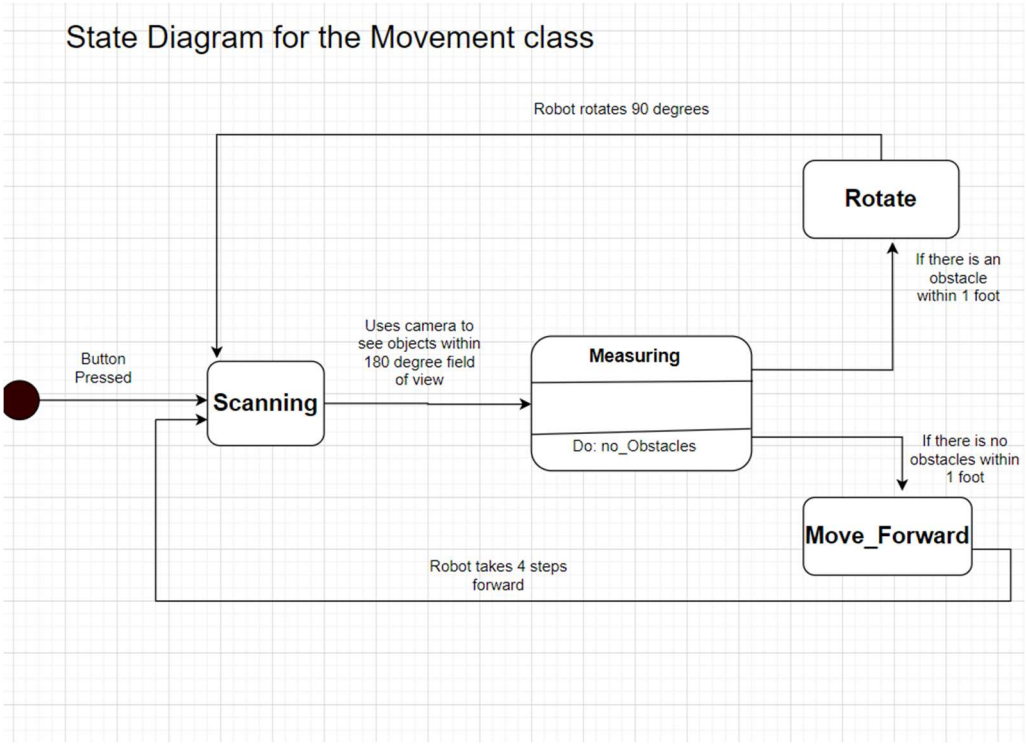


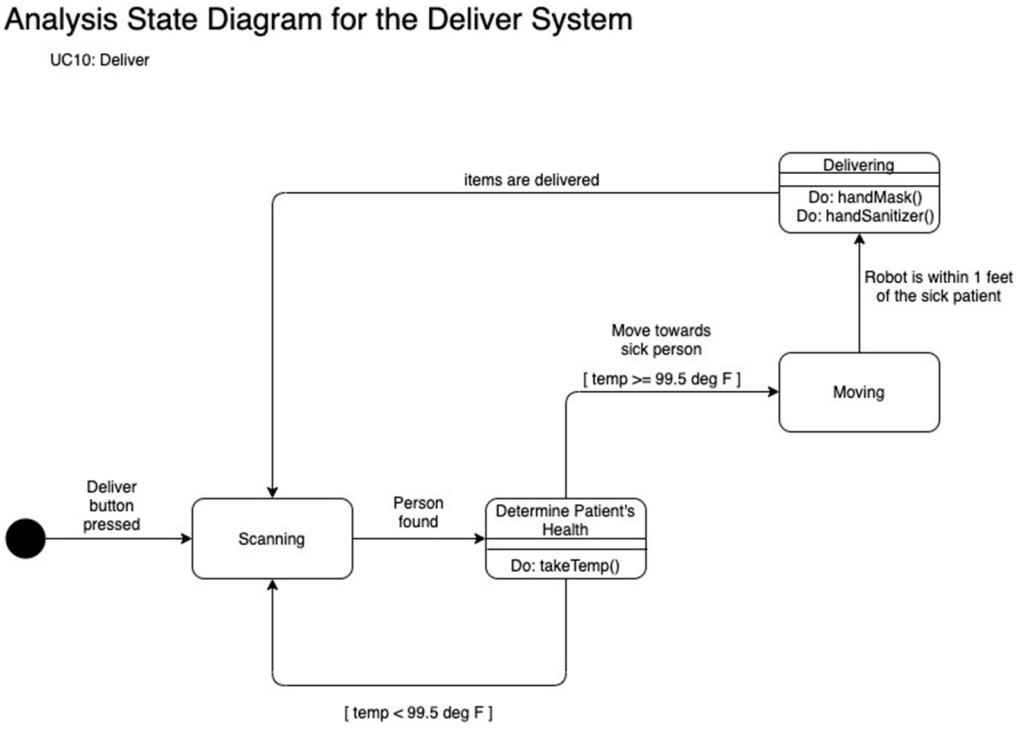
Swimlane Diagram

Swim-lane Diagram: UC10 Deliver



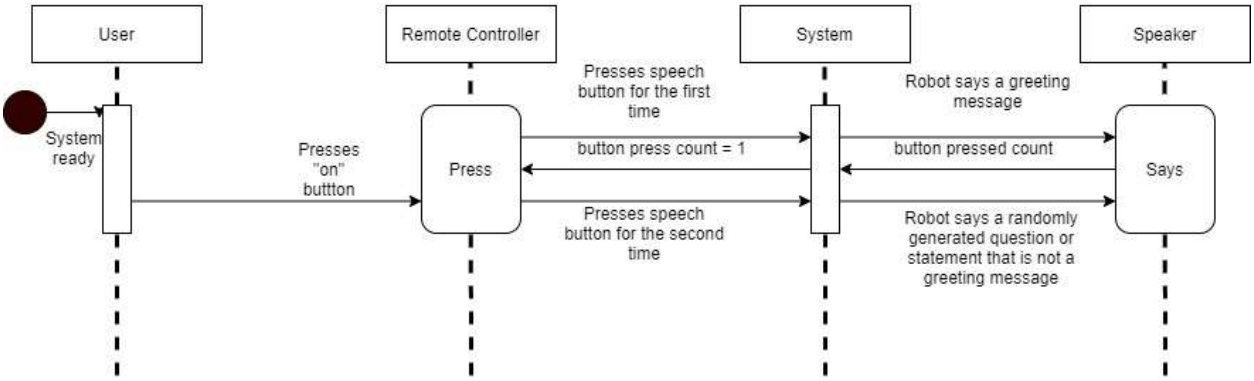
State Diagram





Sequence Diagram

Sequence Diagram for User Case #1



Traceability Matrix

- The traceability matrix is done by looking at all the functional requirements and use cases and seeing which requirement is implemented by which use case.

- This is helpful in the future because it allows us to easily obtain information about the relations between functional requirements and use cases

	FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8	FR9	FR10
UC1	•									
UC2		•								
UC3			•							
UC4				•						
UC5					•					
UC6						•				
UC7	•	•	•	•			•			
UC8	•	•	•	•				•		
UC9	•	•	•	•					•	
UC10	•	•	•	•	•	•				•

Modeling: Design

Architectural Design

Explanation

Architectural Style Decision Explanation:

The architectural style that we have chosen to implement is the layered architecture. fRobot requires a kernel and this was a huge indicator that the layered architecture would be the best choice. The kernel would sit in the core layer. Any abstraction that we add would sit in a layer that is higher than the kernel.

Pros: easy to add features

Cons: if there are too many layers, it would be difficult to manage

The reason why we did not choose to use the data flow architecture is because fRobot does not require complex calculations that would be useful in a data flow architecture. Data flow architectures are also advantageous when you need a series of manipulations on the data which is not required for fRobot as well.

Pros: complex calculations

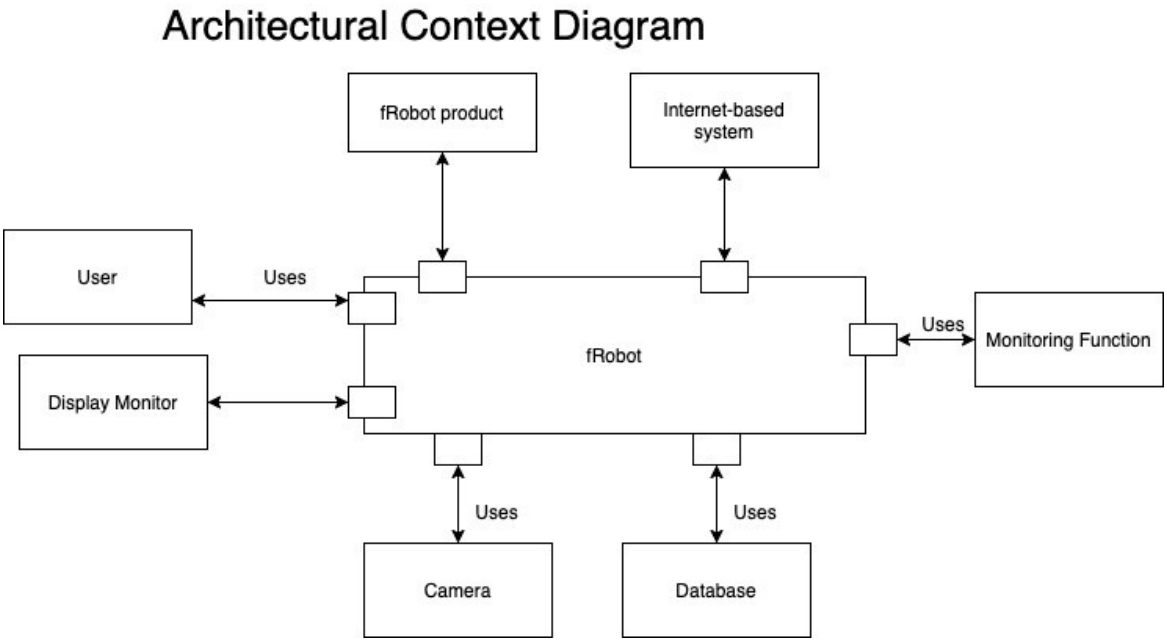
Cons: data flows in one direction only

Data centered architectures is a style of architecture where the data repository sits in the middle and all the components would surround this. This is an easy style to implement, however, there's no (or very little) communication between the components. This would make it hard to design a system where all the components would be able to implicitly communicate with the kernel through several layers.

Pros: easy to add components

Cons: difficult to make the components talk to each other

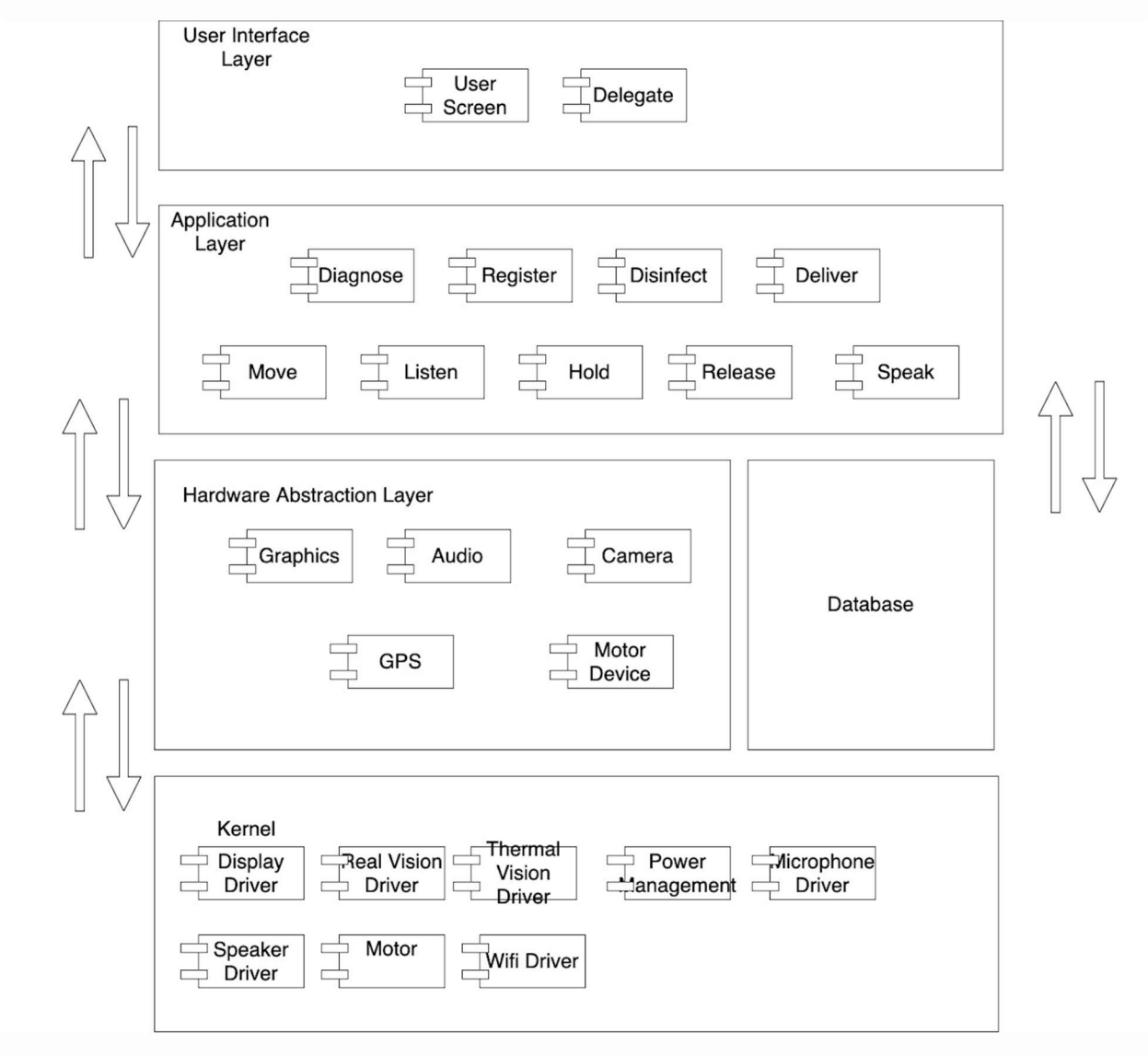
Architectural Context Diagram



Explanation:

fRobot is a system that is connected to the product and the internet shown above. To the left of the system, we have two entities that will feed data to fRobot. These are the user and the display monitor. In a future release, we will add the monitoring function that will use the fRobot system to monitor patients in the hospital. fRobot is a system that uses a camera and a database to accomplish its needs.

Overall Architectural Structure

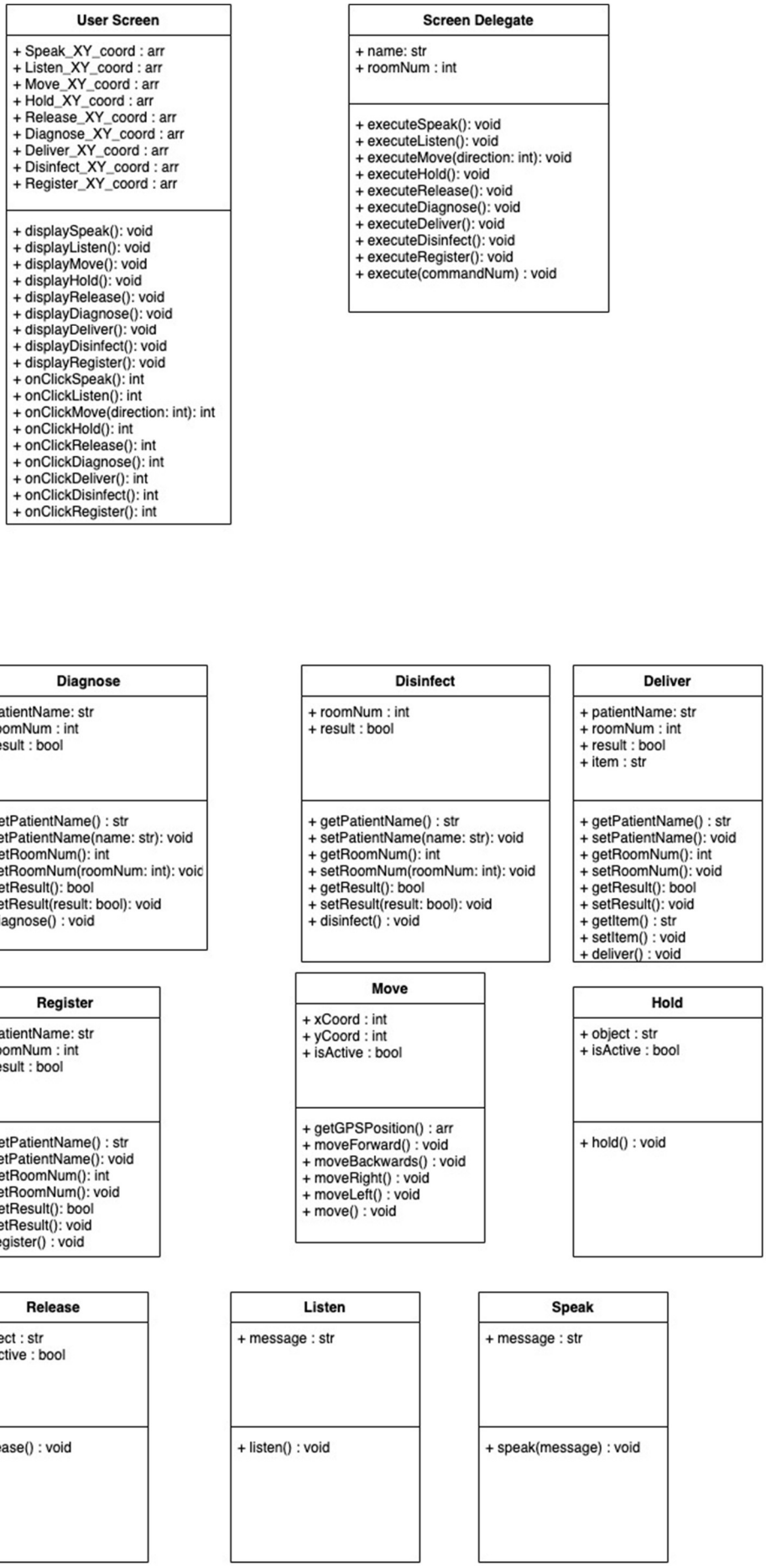


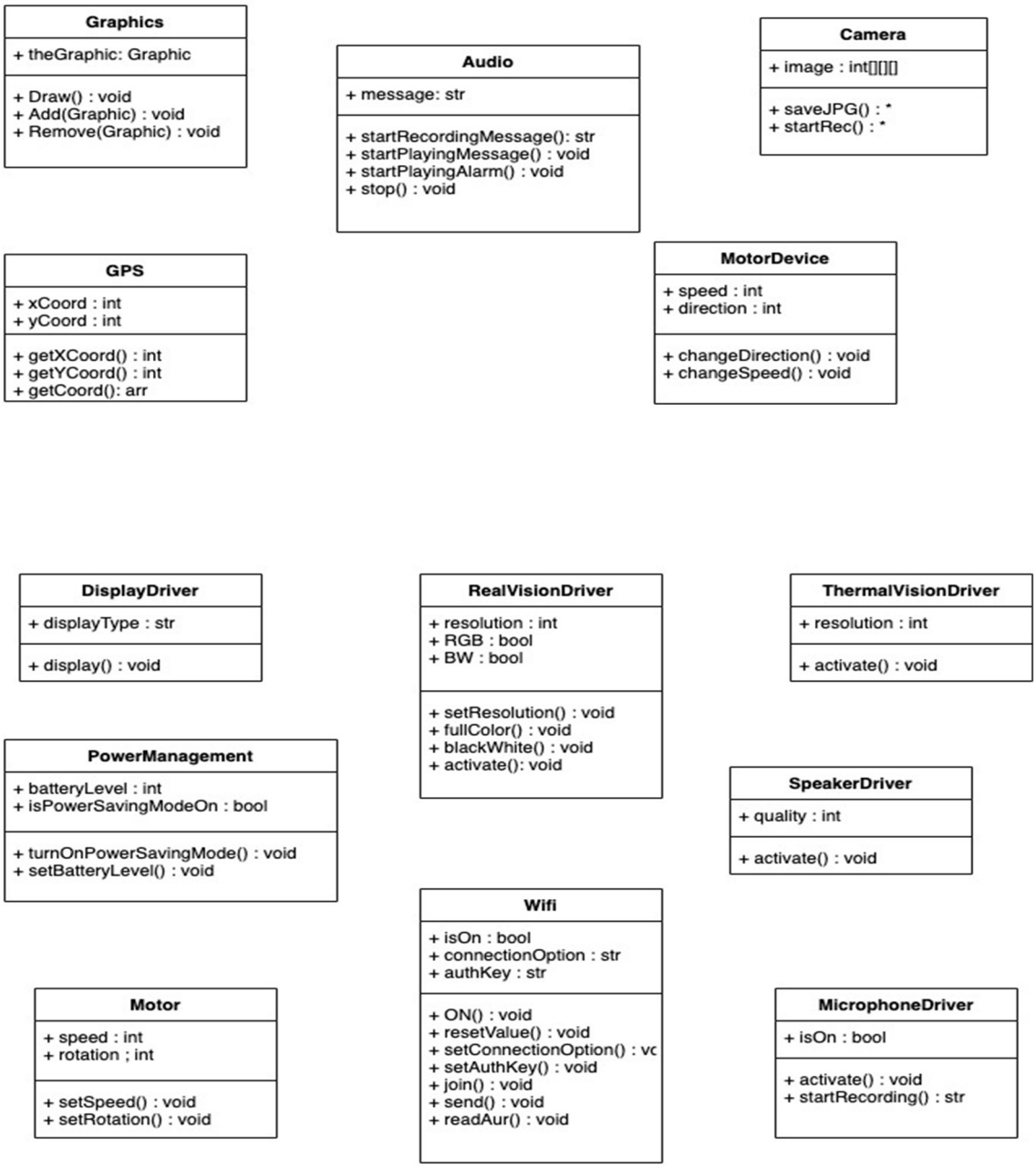
Explanation:

The layered architecture allows us to build abstractions on top of abstractions. At the kernel level, we have all the device drivers that allows us to manipulate the devices directly. In the hardware abstraction layer, we have abstracted the hardware to make it easy for developers to operate the system. The application layer contains components that will do the business logic for the system. At the user interface layer, we will abstract everything and allow the user to perform any of the functions that fRobot is capable of.

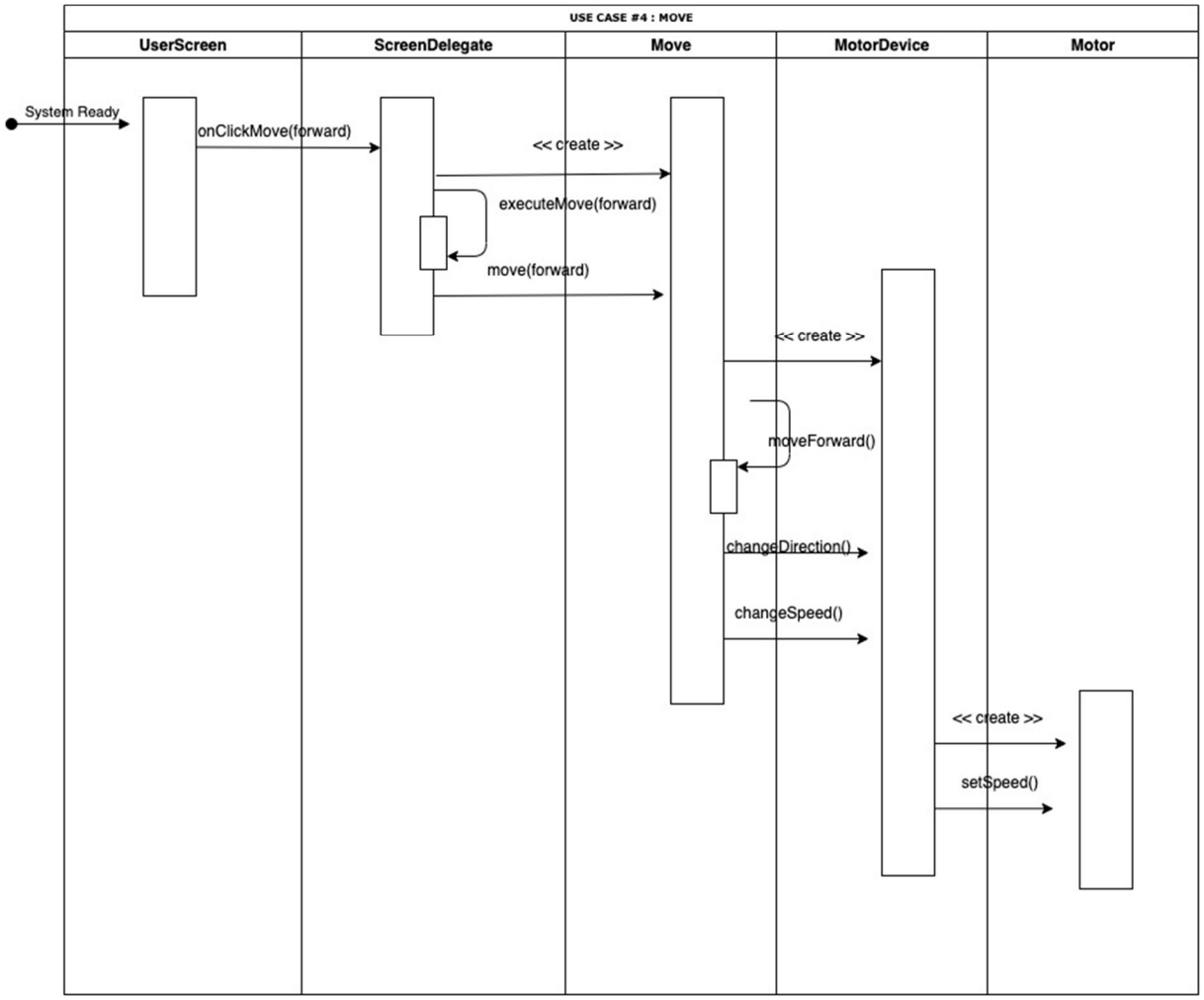
Component-Level Design

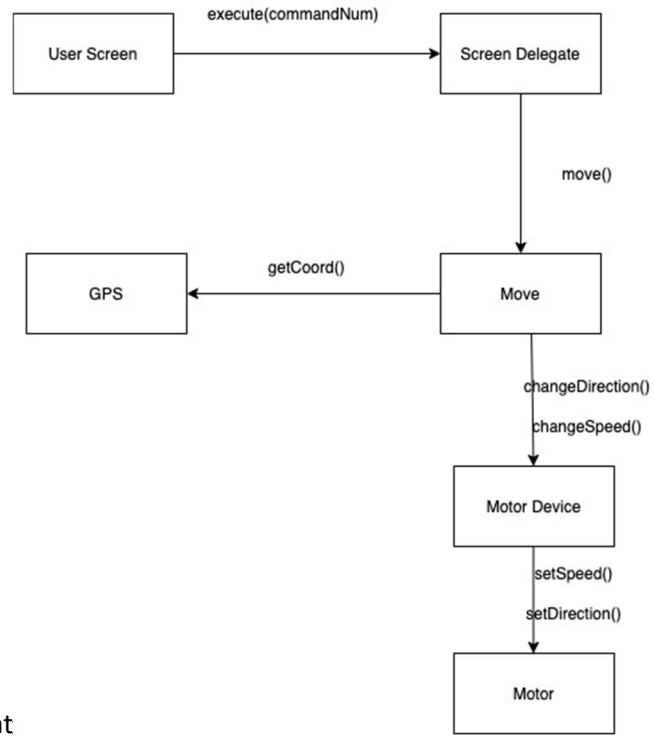
Design Class Diagram


User Screen
+ Speak_XY_coord : arr
+ Listen_XY_coord : arr
+ Move_XY_coord : arr
+ Hold_XY_coord : arr
+ Release_XY_coord : arr
+ Diagnose_XY_coord : arr
+ Deliver_XY_coord : arr
+ Disinfect_XY_coord : arr
+ Register_XY_coord : arr
+ displaySpeak(): void
+ displayListen(): void
+ displayMove(): void
+ displayHold(): void
+ displayRelease(): void
+ displayDiagnose(): void
+ displayDeliver(): void
+ displayDisinfect(): void
+ displayRegister(): void
+ onClickSpeak(): int
+ onClickListen(): int
+ onClickMove(direction: int): int
+ onClickHold(): int
+ onClickRelease(): int
+ onClickDiagnose(): int
+ onClickDeliver(): int
+ onClickDisinfect(): int
+ onClickRegister(): int
Screen Delegate
+ name: str
+ roomNum : int
+ executeSpeak(): void
+ executeListen(): void
+ executeMove(direction: int): void
+ executeHold(): void
+ executeRelease(): void
+ executeDiagnose(): void
+ executeDeliver(): void
+ executeDisinfect(): void
+ executeRegister(): void
+ execute(commandNum) : void
Diagnose
+ patientName: str
+ roomNum : int
+ result : bool
+ getPatientName() : str
+ setPatientName(name: str): void
+ getRoomNum(): int
+ setRoomNum(roomNum: int): voic
+ getResult(): bool
+ setResult(result: bool): void
+ diagnose() : void
Disinfect
+ roomNum : int
+ result : bool
+ getPatientName() : str
+ setPatientName(name: str): void
+ getRoomNum(): int
+ setRoomNum(roomNum: int): void
+ getResult(): bool
+ setResult(result: bool): void
+ disinfect() : void
Deliver
+ patientName: str
+ roomNum : int
+ result : bool
+ item : str
+ getPatientName() : str
+ setPatientName(): void
+ getRoomNum(): int
+ setRoomNum(): void
+ getResult(): bool
+ setResult(): void
+ getItem() : str
+ setItem() : void
+ deliver() : void
Register
+ patientName: str
+ roomNum : int
+ result : bool
+ getPatientName() : str
+ setPatientName(): void
+ getRoomNum(): int
+ setRoomNum(): void
+ getResult(): bool
+ setResult(): void
+ register() : void
Move
+ xCoord : int
+ yCoord : int
+ isActive : bool
+ getGPSPosition() : arr
+ moveForward() : void
+ moveBackwards() : void
+ moveRight() : void
+ moveLeft() : void
+ move() : void
Hold
+ object : str
+ isActive : bool
+ hold() : void
Release
+ object : str
+ isActive : bool
+ release() : void
Listen
+ message : str
+ listen() : void
Speak
+ message : str
+ speak(message) : void


Graphics
+ theGraphic: Graphic
+ Draw() : void
+ Add(Graphic) : void
+ Remove(Graphic) : void
Audio
+ message: str
+ startRecordingMessage(): str
+ startPlayingMessage() : void
+ startPlayingAlarm() : void
+ stop() : void
Camera
+ image : int[][][]
+ saveJPG() : *
+ startRec() : *
GPS
+ xCoord : int
+ yCoord : int
+ getXCoord() : int
+ getYCoord() : int
+ getCoord(): arr
MotorDevice
+ speed : int
+ direction : int
+ changeDirection() : void
+ changeSpeed() : void
DisplayDriver
+ displayType : str
+ display() : void
RealVisionDriver
+ resolution : int
+ RGB : bool
+ BW : bool
+ setResolution() : void
+ fullColor() : void
+ blackWhite() : void
+ activate(): void
ThermalVisionDriver
+ resolution : int
+ activate() : void
PowerManagement
+ batteryLevel : int
+ isPowerSavingModeOn : bool
+ turnOnPowerSavingMode() : void
+ setBatteryLevel() : void
SpeakerDriver
+ quality : int
+ activate() : void
Wifi
+ isOn : bool
+ connectionOption : str
+ authKey : str
+ ON() : void
+ resetValue() : void
+ setConnectionOption() : vc
+ setAuthKey() : void
+ join() : void
+ send() : void
+ readAur() : void
Motor
+ speed : int
+ rotation ; int
+ setSpeed() : void
+ setRotation() : void
MicrophoneDriver
+ isOn : bool
+ activate() : void
+ startRecording() : str

Interaction Diagrams

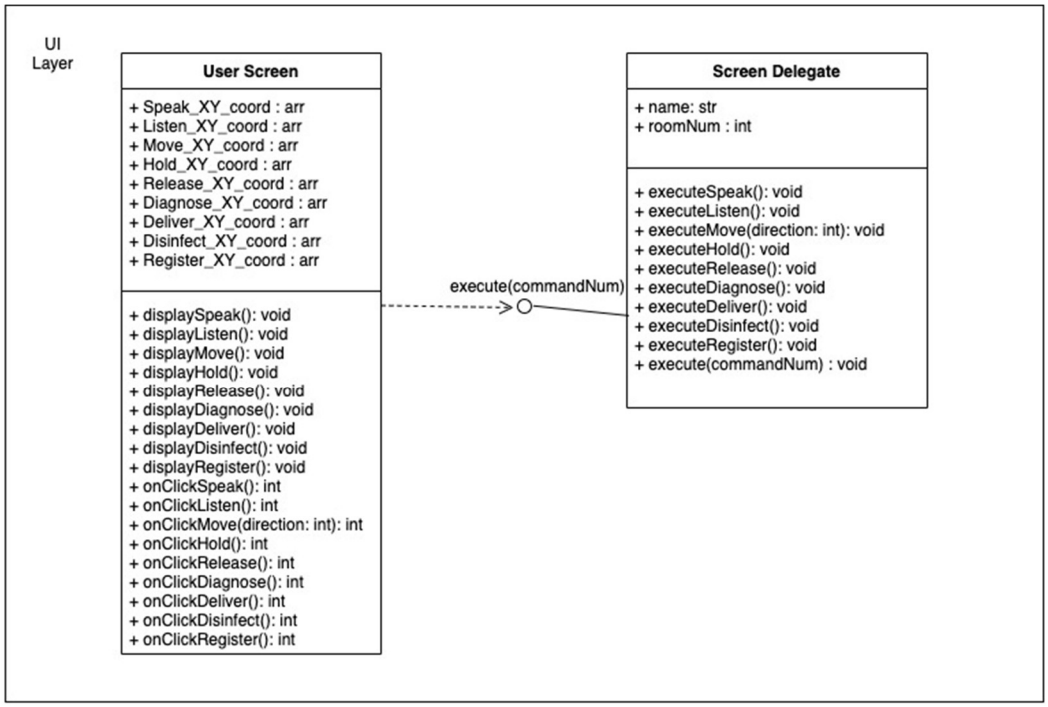


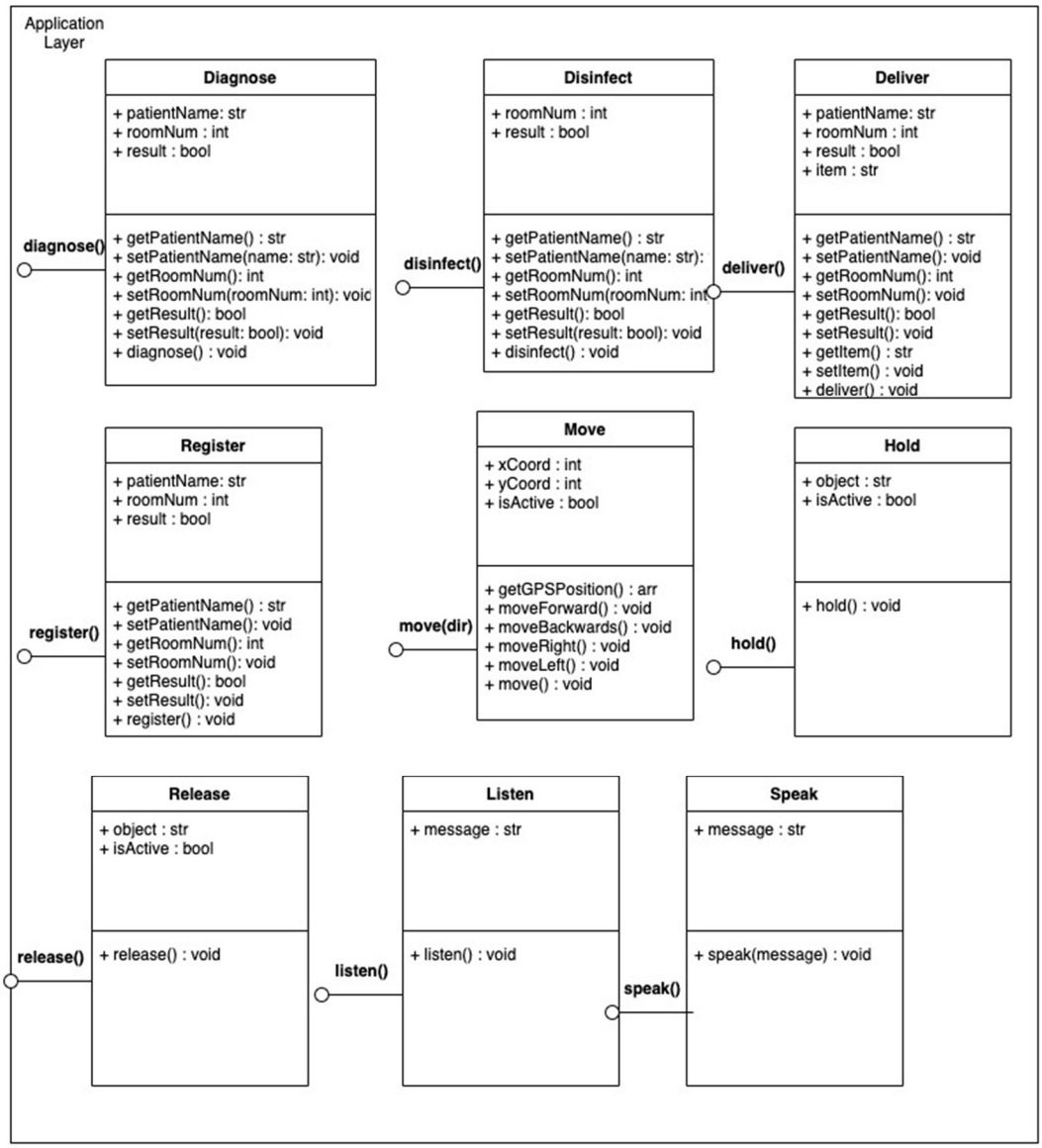


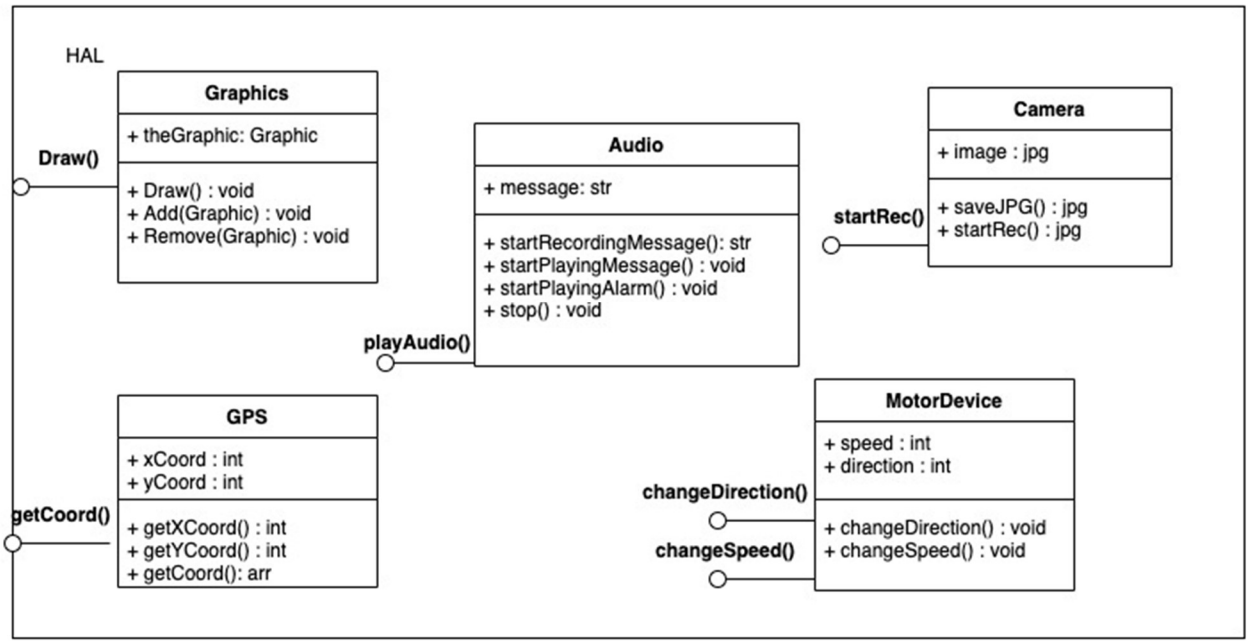
Use case #4: Movement

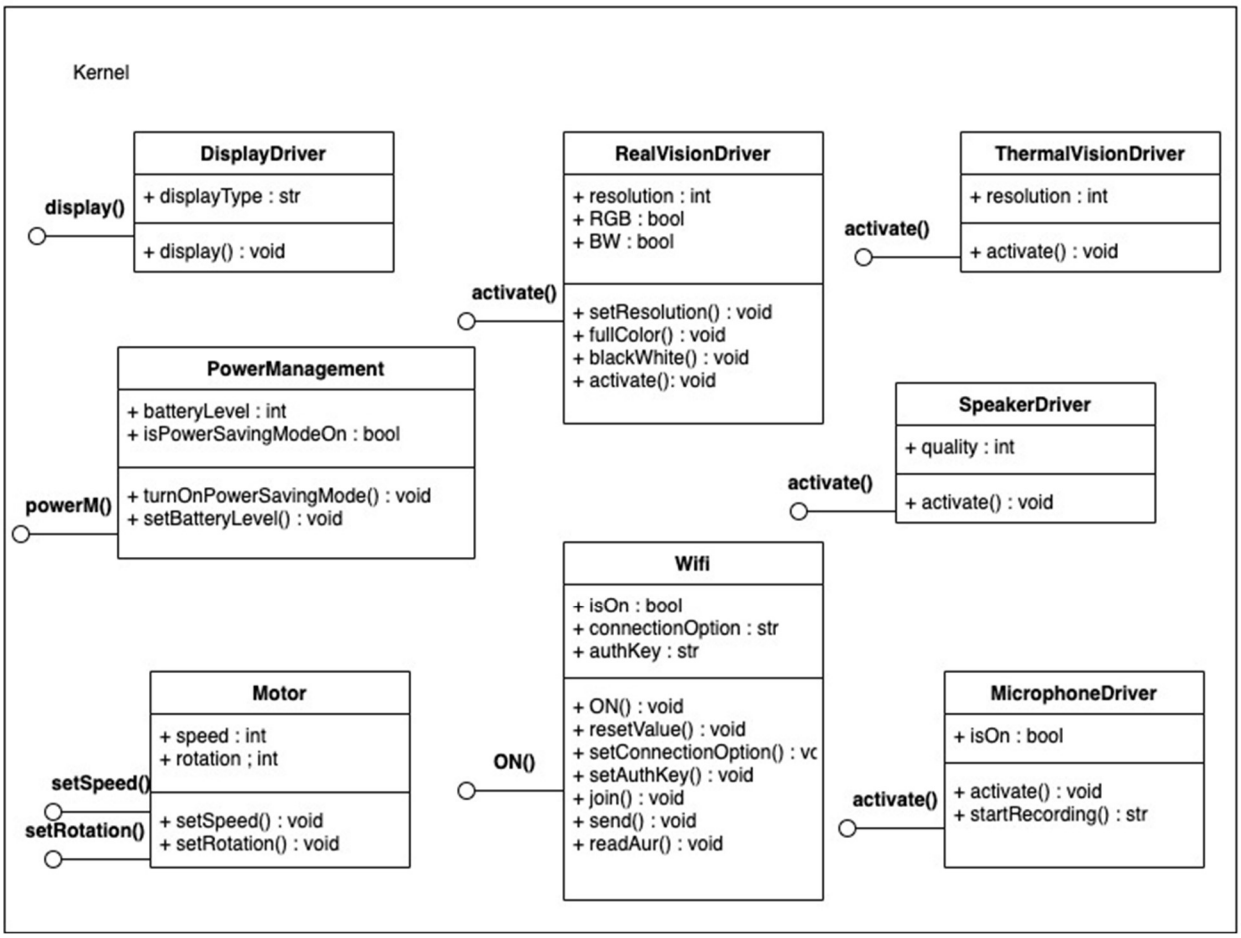
Collaboration / Communication Diagram

Refined Architectural Structure Diagram

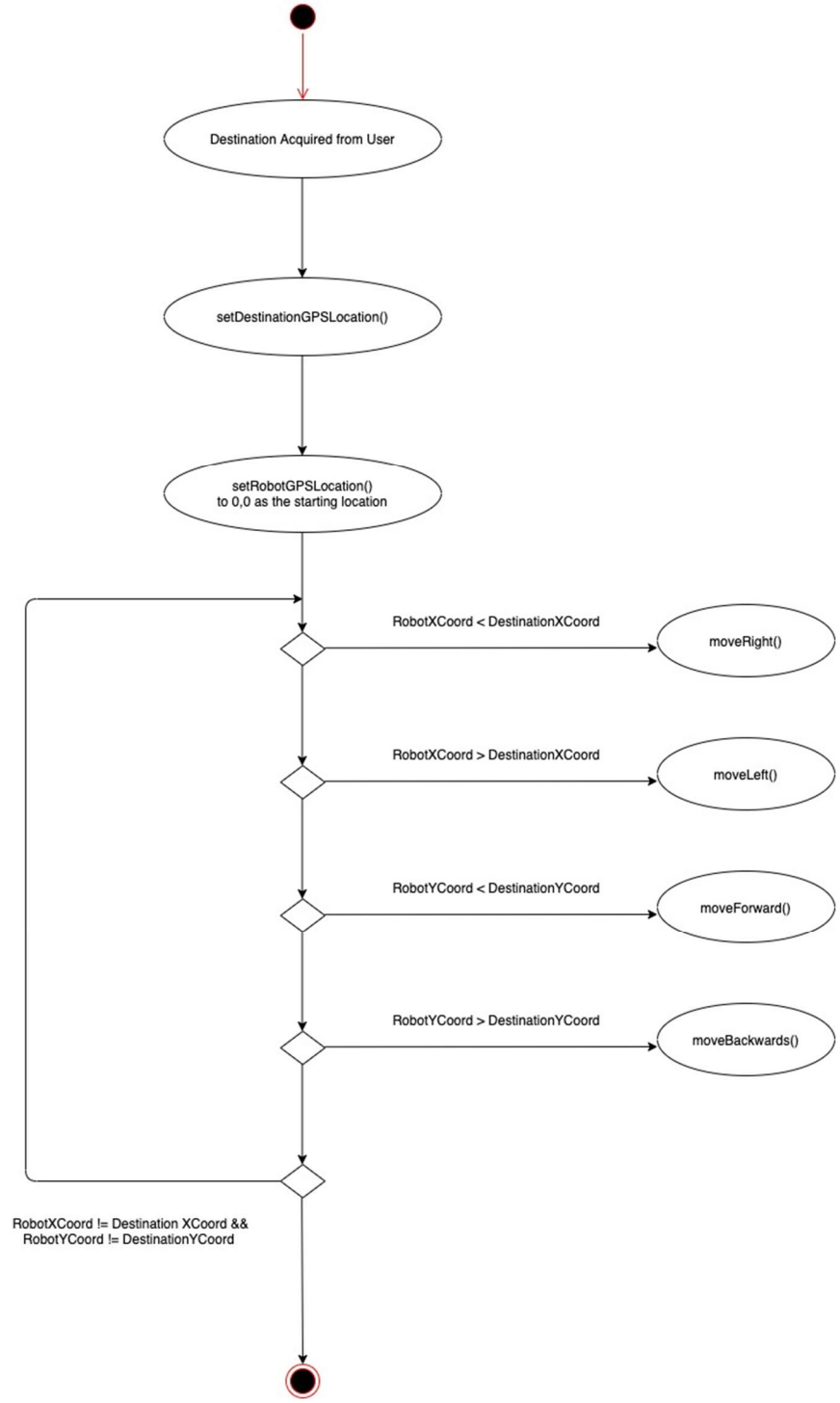

UI Layer
User Screen
+ Speak_XY_coord : arr
+ Listen_XY_coord : arr
+ Move_XY_coord : arr
+ Hold_XY_coord : arr
+ Release_XY_coord : arr
+ Diagnose_XY_coord : arr
+ Deliver_XY_coord : arr
+ Disinfect_XY_coord : arr
+ Register_XY_coord : arr
+ displaySpeak(): void
+ displayListen(): void
+ displayMove(): void
+ displayHold(): void
+ displayRelease(): void
+ displayDiagnose(): void
+ displayDeliver(): void
+ displayDisinfect(): void
+ displayRegister(): void
+ onClickSpeak(): int
+ onClickListen(): int
+ onClickMove(direction: int): int
+ onClickHold(): int
+ onClickRelease(): int
+ onClickDiagnose(): int
+ onClickDeliver(): int
+ onClickDisinfect(): int
+ onClickRegister(): int
execute(commandNum)
Screen Delegate
+ name: str
+ roomNum : int
+ executeSpeak(): void
+ executeListen(): void
+ executeMove(direction: int): void
+ executeHold(): void
+ executeRelease(): void
+ executeDiagnose(): void
+ executeDeliver(): void
+ executeDisinfect(): void
+ executeRegister(): void
+ execute(commandNum) : void


Application Layer
Diagnose
+ patientName: str
+ roomNum : int
+ result : bool
+ getPatientName() : str
+ setPatientName(name: str): void
+ getRoomNum(): int
+ setRoomNum(roomNum: int): voic
+ getResult(): bool
+ setResult(result: bool): void
+ diagnose() : void
diagnose()
Disinfect
+ roomNum : int
+ result : bool
+ getPatientName() : str
+ setPatientName(name: str):
+ getRoomNum(): int
+ setRoomNum(roomNum: int
+ getResult(): bool
+ setResult(result: bool): void
+ disinfect() : void
disinfect()
Deliver
+ patientName: str
+ roomNum : int
+ result : bool
+ item : str
+ getPatientName() : str
+ setPatientName(): void
+ getRoomNum(): int
+ setRoomNum(): void
+ getResult(): bool
+ setResult(): void
+ getItem() : str
+ setItem() : void
+ deliver() : void
deliver()
Register
+ patientName: str
+ roomNum : int
+ result : bool
+ getPatientName() : str
+ setPatientName(): void
+ getRoomNum(): int
+ setRoomNum(): void
+ getResult(): bool
+ setResult(): void
+ register() : void
register()
Move
+ xCoord : int
+ yCoord : int
+ isActive : bool
+ getGPSPosition() : arr
+ moveForward() : void
+ moveBackwards() : void
+ moveRight() : void
+ moveLeft() : void
+ move() : void
move(dir)
Hold
+ object : str
+ isActive : bool
+ hold() : void
hold()
Release
+ object : str
+ isActive : bool
+ release() : void
release()
Listen
+ message : str
+ listen() : void
listen()
Speak
+ message : str
+ speak(message) : void
speak()

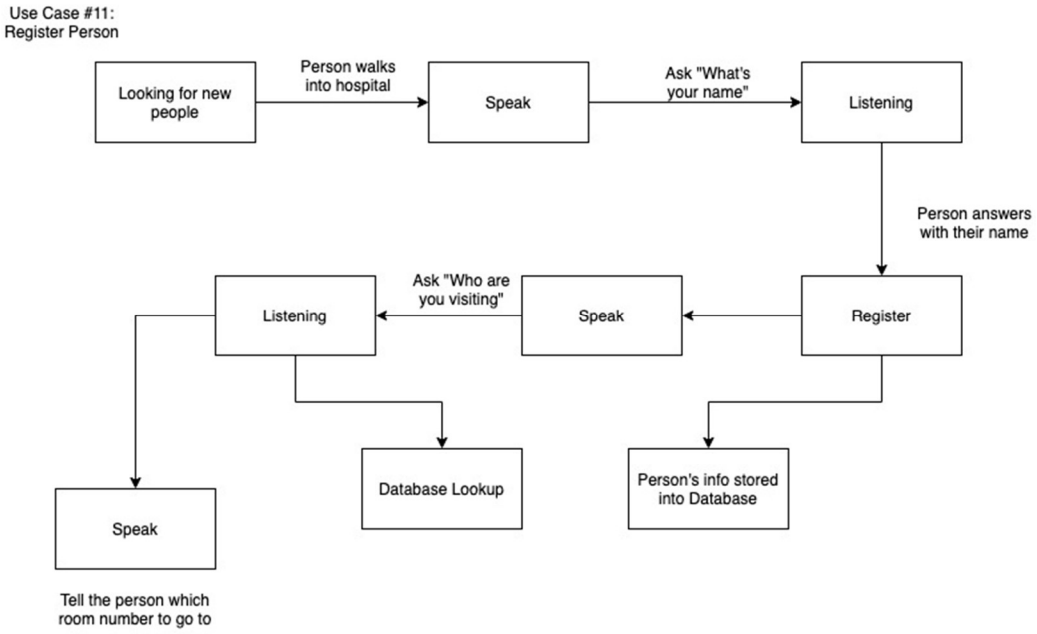

HAL
Graphics
+ theGraphic: Graphic
+ Draw() : void
+ Add(Graphic) : void
+ Remove(Graphic) : void
Draw()
Audio
+ message: str
+ startRecordingMessage(): str
+ startPlayingMessage() : void
+ startPlayingAlarm() : void
+ stop() : void
playAudio()
Camera
+ image : jpg
+ saveJPG() : jpg
+ startRec() : jpg
startRec()
GPS
+ xCoord : int
+ yCoord : int
+ getXCoord() : int
+ getYCoord() : int
+ getCoord(): arr
getCoord()
MotorDevice
+ speed : int
+ direction : int
+ changeDirection() : void
+ changeSpeed() : void
changeDirection()
changeSpeed()


Kernel
DisplayDriver
+ displayType : str
+ display() : void
display()
RealVisionDriver
+ resolution : int
+ RGB : bool
+ BW : bool
+ setResolution() : void
+ fullColor() : void
+ blackWhite() : void
+ activate(): void
activate()
ThermalVisionDriver
+ resolution : int
+ activate() : void
activate()
PowerManagement
+ batteryLevel : int
+ isPowerSavingModeOn : bool
+ turnOnPowerSavingMode() : void
+ setBatteryLevel() : void
powerM()
SpeakerDriver
+ quality : int
+ activate() : void
activate()
Wifi
+ isOn : bool
+ connectionOption : str
+ authKey : str
+ ON() : void
+ resetValue() : void
+ setConnectionOption() : vc
+ setAuthKey() : void
+ join() : void
+ send() : void
+ readAur() : void
ON()
Motor
+ speed : int
+ rotation ; int
+ setSpeed() : void
+ setRotation() : void
setSpeed()
setRotation()
MicrophoneDriver
+ isOn : bool
+ activate() : void
+ startRecording() : str
activate()

Activity Diagram



State Chart



User Experience / Interface Design

Persona

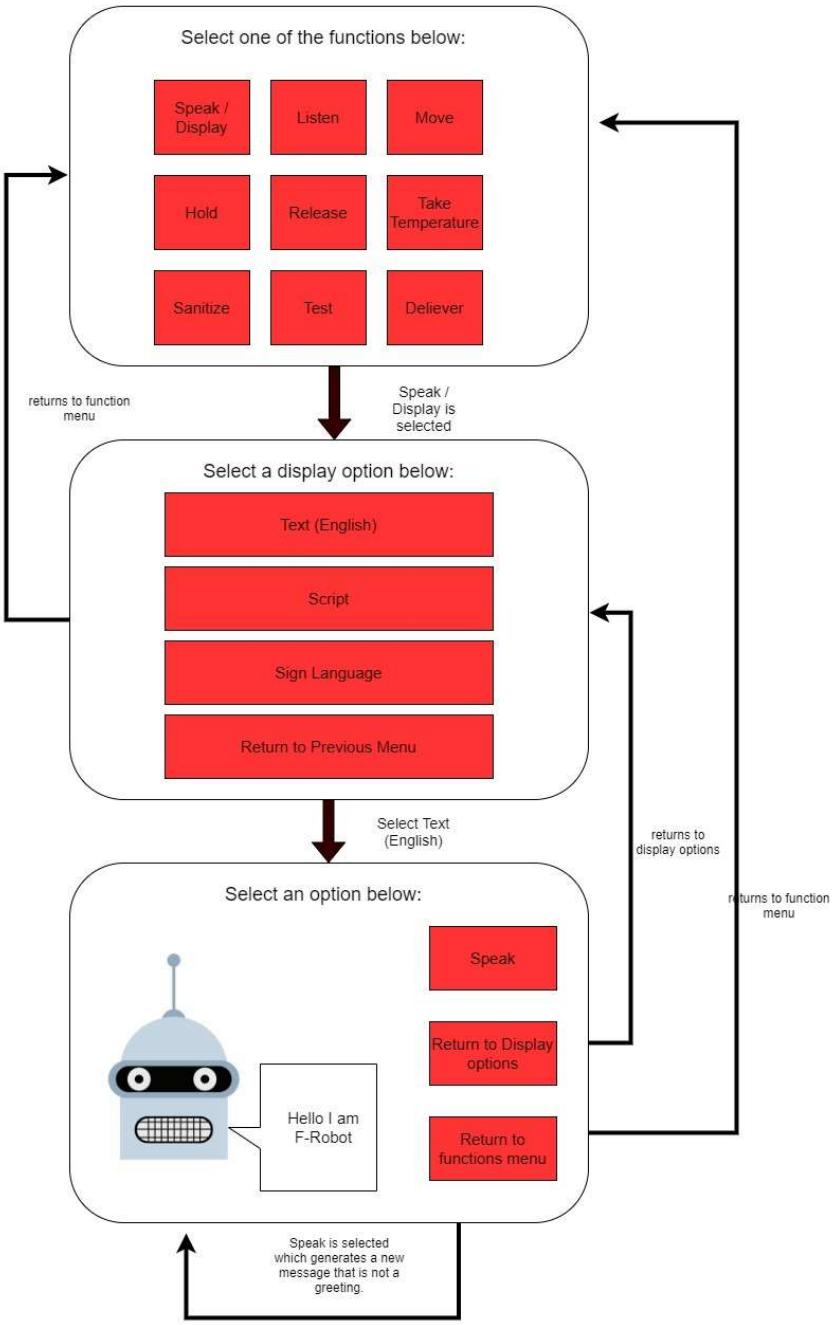
Persona for a Doctor

Dr. Wilks

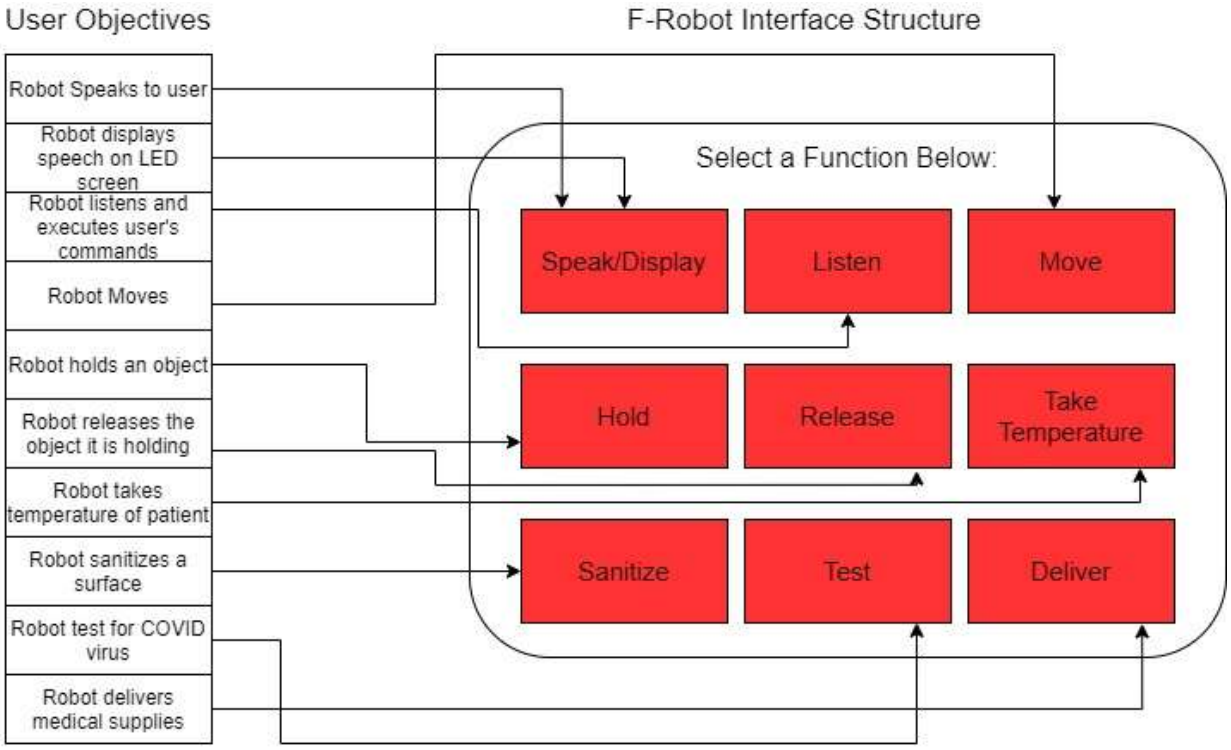
Psycho-graphics	Personal Background	Personal Background	Scenario
-Enjoys helping people that are in need. -Prioritizes safety for himself and his patients. -Emphasizes cleanliness. -Not too used to dealing with newer forms of technology but is willing to put in time to learn. -Does not prefer face-to-face contact when dealing with COVID patients.	Age: 61 Status: Married father of three Education: Doctorate Degree of Medicine	Occupation: Full-time doctor at Hoag Hospital Income: $180,000 per year Education: Doctorate Degree of Medicine	-At least 10 times a day I am called in to either test a patient for corona virus or to deliver medical supplies to someone who is confirmed to be infected with COVID. -Being as though I am an older male, it becomes increasingly dangerous for me to consistently be in contact with such a dangerous virus. -Ideally, it would be in my best interest as well as my fellow medical professional peers to have a robot that handles the testing of patients, the delivery of medical supplies to those who are confirmed infected, and the disinfecting of rooms.
	User Environment	End Goals	
	Location: Hoag Hosipital Devices: FRobot to handle corona virus related patients. As well as his laptop and Iphone.	Use mobile application to control frobot that is designed to be used in a hospital system. The robot will drastically decreases the number of times a medical professional will have to come in direct contact with a potential infected patient.	

Preliminary Screen Layout

Screen Layout for User Case #01 & #02



Menu/Navigation



Explanation: From the Use Cases and Functional Requirements we constructed the user objectives and then simply mapped the objectives to interface structure menu buttons.

Mobile App Design

13 Mobile UI Design Considerations / Mobility Design Best Practices

Interface Design Considerations

- User Interface Brand Signatures:
    - Deploy this app/robot in a hospital
        - Make it easy to use by nurses and doctors
    - Test for coronavirus
    - Take patient's temperature with thermal imaging camera
    - Disinfect area with ultraviolet light
    - Deliver medical materials to patients
    - Allow voice commands to control the robot
    - Facial recognition for easy access

- Focus the portfolio of products:

    - Target platforms will include anything having to do with medical professionals and the integration with any software that they use.
    - Apple and Android

- Core User Stories:
    - As a patient, I want the robot to give accurate corona virus test results so that I know people around me and myself are safe.
    - As a doctor, I want the robot to truly disinfect an area so I do not put my patients in jeopardy.
    - As a stakeholder, I want all of the robot's functions to work properly and in a timely matter so that there are no negative reviews.

- Optimize user interface flows and elements:
    - Make every element obvious with placement, color, and size of element.
    - Build interfaces only if there is a plan to generate an interaction.
    - Do not be repetitive with interfaces, keep interfaces concise.
    - Make each interface flow effortlessly into the next.
    - Flow from beginning of interactions to end without any interruptions.

- Scaling Rules:
    - Allow a maximum of three buttons to be displayed on the screen at once
    - Don't go over 150 characters per page
    - One picture per page

- User performance dashboard:
    - Dashboard will show the number of features implemented
    - Display the number of users

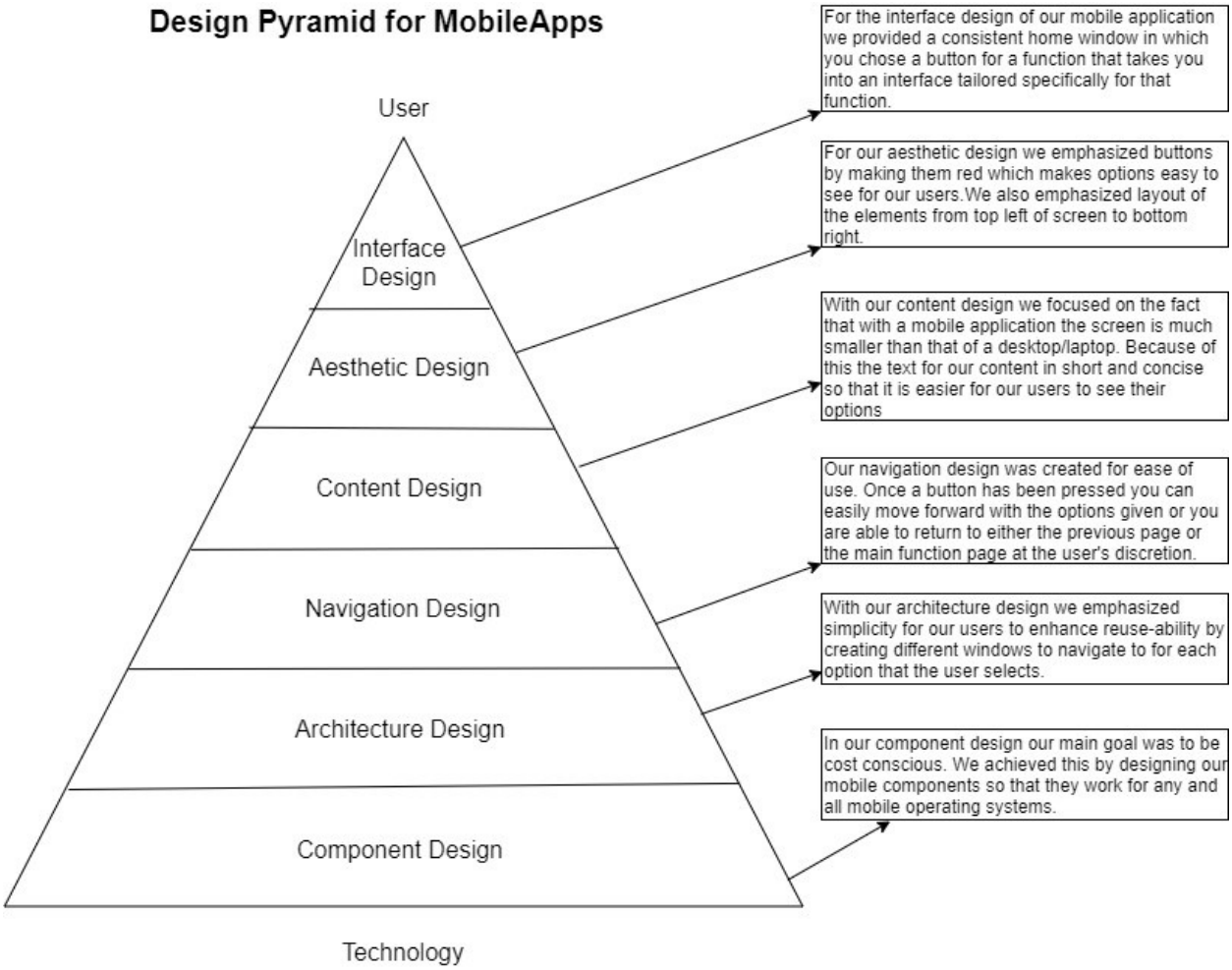
- Champion-dedicated user interface engineering skills:
    - Preload all information to allow smooth transitions between screens

Mobility Design Best Practices

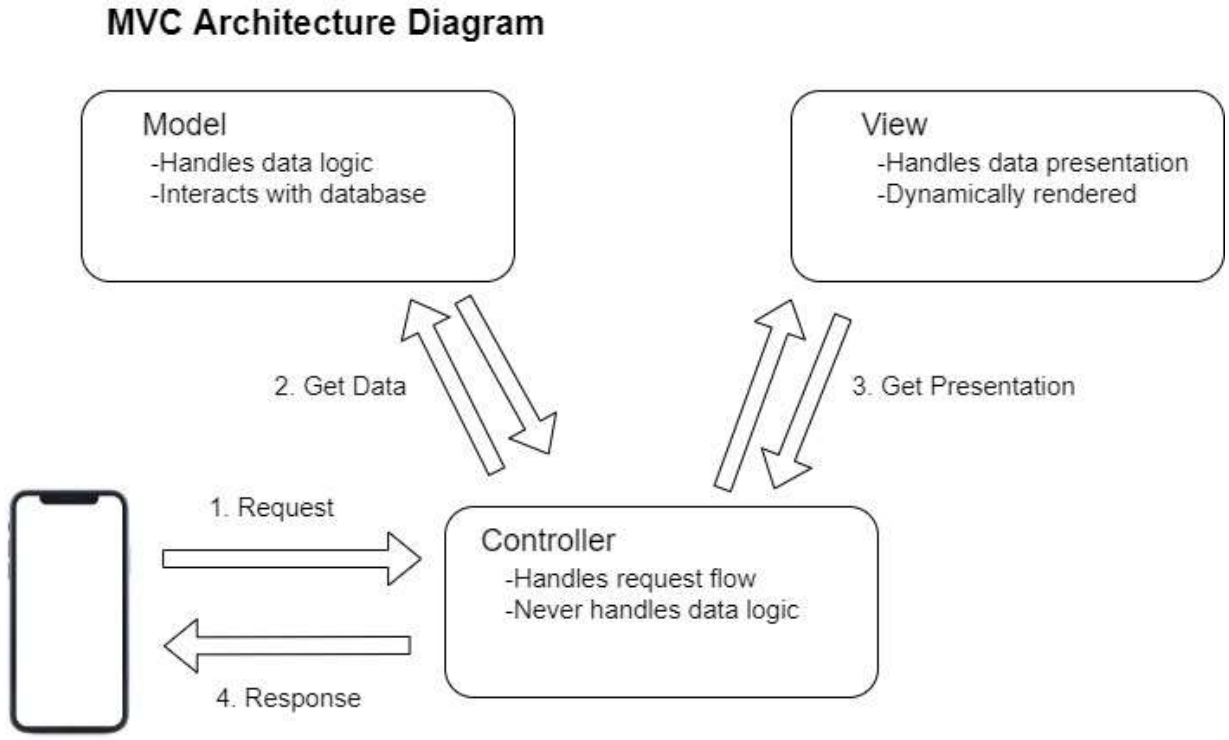
- Identify your audience:
    - Nurses, doctors
- Design for context of use:
    - Working in a hospital
- There is a fine line between simplicity and laziness:
    - Must identify the core user stories and allow the user to successfully initiate these stories from every page of the app
- Use the platform as an advantage:
    - Mobile app will be designed for Apple devices. Use Apple's geolocation feature as well as facial recognition feature

- Make scrollbars and selection highlighting more salient:
    - Use colors that are on opposite ends of the color wheel so that changes are more noticeable
- Increase discoverability of advanced functionality:
    - At the bottom of the screen, there will be a row of hot keys
- Clever icons should never be developed at the expense of user understanding:
    - Include text at the bottom of icons
- Support user expectations for personalization:
    - Users should be allowed to personalize the number of rooms that are in the hospital, the number of patients. Also, this app shall allow the addition of new users
- Long scrolling forms trump multiple screens on mobile devices
    - Keep everything on one page

Design Pyramid for MobileApps



MVC Architecture



Describe what M, V, and C mean with your system design (all within 1-2 pages).

Explanation in terms of UC01 & UC02: The Control will receive the request of the speech and display speech functions from the user. This request is then sent to the model which will access the database and get a speech dialog which is then sent back to the controller. This dialog is then sent to the view section where it will be converted to either text, script, or sign language. This output is then sent back to the controller and in turn sent back to the user for the desired output.

Pattern-Based Design

Design problem:

The design problem that we have is to figure out how to implement all the commands. The commands that fRobot must take action on are: move, hold, release, listen, speak, deliver, diagnose, disinfect, and register.

Design pattern considerations:

Command pattern is a behavioral design pattern. An object is used to encapsulate all the information required to perform an action. This is perfect for our fRobot. For example, our register command will need to ask the person if they are a new patient, a returning patient, or a visitor. It will then inquire the

person for their name and store it in the database. If the person who just entered the hospital was a new patient, fRobot will then tell the new patient to wait in the room that has been disinfected. fRobot will look into its database and make sure that the room has been disinfected earlier. All of these actions (and more) are required to perform a command, so it is helpful to store them as an object. Hence, the reason why we chose to use the command pattern.

The composite pattern is a design pattern that is a group of objects that are treated the same way as a single instance of the same type of object.

Pattern Organizing Table

	Database	Application	Implementation	Infrastructure
Data / Content				
Set Patient Info.	Composite Pattern		Façade Pattern	Composite Pattern
Display all Patients	Façade pattern		Façade pattern	Façade pattern
Architecture				
Commands		Command Pattern	Command Pattern	Command Pattern
Component-level				
User Interface				
Cross device browser platform				Bridge Pattern

Construction: Code

Protoype Code from Design Class Diagram

UI Layer

```
class UserScreen
{
public:
	void displaySpeak();
	void displayListen();
	void displayMove();
	void displayHold();
	void displayRelease();
	void displayDiagnose();
```

```
        void displayDeliver();
        void displayDisinfect();
        void displayRegister();
        int onClickSpeak();
        int onClickListen();
        int onClickMove(int direction);
        int onClickHold();
        int onClickRelease();
        int onClickDiagnose();
        int onClickDeliver();
        int onClickDisinfect();
        int onClickRegister();

private:
        int speak_XY_coord[2];
        int listen_XY_coord[2];
        int move_XY_coord[2];
        int hold_XY_coord[2];
        int release_XY_coord[2];
        int diagnose_XY_coord[2];
        int deliver_XY_coord[2];
        int disinfect_XY_coord[2];
        int register_XY_coord[2];
};


class ScreenDelegate
{
public:
        void executeSpeak();
        void executeListen();
        void executeMove(int direction);
        void executeHold();
        void executeRelease();
        void executeDiagnose();
        void executeDeliver();
        void executeDisinfect();
        void executeRegister();
        void execute(int commandNum);

private:
        string name;
        int roomNum;
};
```

Application Layer

```
class Diagnose
{
public:
        string getPatientName();
        void setPatientName(string name);
        int getRommNum();
        void setRommNum(int num);
        bool getResult();
```

```
        void setResult(bool newResult);
        void diagnose();

private:
        string patientName;
        int roomNum;
        bool result;
};

class Disinfect
{
public:
        int getRommNum();
        void setRommNum(int num);
        bool getisDisinfected();
        void setisDisinfected(bool newResult);
        void disinfect();

private:
        int roomNum;
        bool isDisinfected;
};

class Deliver
{
public:
        string getPatientName();
        void setPatientName(string name);
        int getRommNum();
        void setRommNum(int num);
        bool getResult();
        void setResult(bool newResult);
        string getItem();
        void setItem(string newItem);
        void deliver();

private:
        string patientName;
        int roomNum;
        bool result;
        string item;
};

class Register
{
public:
        string getPatientName();
        void setPatientName(string name);
        int getRommNum();
        void setRommNum(int num);
        bool getResult();
        void setResult(bool newResult);
        void reg();

private:
        string patientName;
        int roomNum;
        bool result;
```

```
};

class Move {
public:
    int* getDestinationGPSPosition() {
        return destinationCoord;
    }
    void setDestinationGPSPosition(int newXCoord, int newYCoord) {
        destinationCoord[0] = newXCoord;
        destinationCoord[1] = newYCoord;
    }
    int* getRobotGPSPosition() {
        return robotCoord;
    }
    void setRobotGPSPosition(int newXCoord, int newYCoord) {
        robotCoord[0] = newXCoord;
        robotCoord[1] = newYCoord;
    }
    void moveForward() {
        robotCoord[1] += 1;
    }
    void moveBackward() {
        robotCoord[1] -= 1;
    }
    void moveRight() {
        robotCoord[0] += 1;
    }
    void moveLeft() {
        robotCoord[0] -= 1;
    }
    void move() {
        int flag = 0;
        while (robotCoord[0] != destinationCoord[0] || robotCoord[1] !=
destinationCoord[1]) {
            if (robotCoord[0] < destinationCoord[0]) {
                moveRight();
            }
            if (robotCoord[0] > destinationCoord[0]) {
                moveLeft();
            }
            if (robotCoord[1] < destinationCoord[1]) {
                moveForward();
            }
            if (robotCoord[1] > destinationCoord[1]) {
                moveBackward();
            }
            flag = 1;
        }
        if (flag == 1) {
            cout << "\nRobot is walking towards location..." << endl;
        }
    }

private:
    int* destinationCoord = new int[2];
    int* robotCoord = new int[2];
    bool isActive;
};
```

```
class Hold
{
public:
	void hold();

private:
	string object;
	bool isActive;
};

class Hold
{
public:
	void release();

private:
	string object;
	bool isActive;
};

class Listen
{
public:
	void listen();

private:
	string message;
};

class Speak
{
public:
	void speak(string mes);

private:
	string message;
};
```

HAL

```
class Graphics
{
public:
	void draw();
	void add(Graphics graphic);
	void remove(Graphics graphic);

private:
	Graphics theGraphic;
};

class Audio
```

```
{
public:
	string startRecordingMessage();
	void startPlayingMessage();
	void startPlayingAlarm();
	void stop();

private:
	string message;
};

class Camera
{
public:
	int* saveJPG();
	int* startRec();

private:
	int* image;
};

class GPS
{
public:
	int getXCoord();
	int getyCoord();
	int* getCoord();

private:
	int xCoord;
	int yCoord;
};

class MotorDevice
{
public:
	void changeDirection();
	void changeSpeed();

private:
	int speed;
	int direction;
};
```

Kernel

```
class DisplayDriver
{
public:
	void display();

private:
	string displayType;
```

```
};

class RealVisionDriver
{
public:
	void setResolution(int newRes);
	void fullColor();
	void blackWhite();
	void activate();

private:
	int resolution;
	bool RGB;
	bool BW;
};

class ThermalVisionDriver
{
public:
	void activate();

private:
	int resolution;
};

class PowerManagement
{
public:
	void turnOnPowerSavingMode();
	void setBatteryLevel(int newBattLevel);

private:
	int batteryLevel;
	bool isPowerSavingModeOn;
};

class SpeakerDriver
{
public:
	void activate();

private:
	int quality;
};

class Motor
{
public:
	void setSpeed(int newSpeed);
	void setRotation(int newRotation);

private:
	int speed;
	int rotation;
};

class Wifi
{
```

```
public:
	void on();
	void resetValue();
	void setCOnnectionOption(string newOption);
	void setAuthKey(string newKey);
	void join();
	void send();
	void readAur();

private:
	bool isOn;
	string connectionOption;
	string authKey;
};

class MicrophoneDriver
{
public:
	void activate();
	string startRecording();

private:
	bool isOn;
};

int main() {
    Move myRobot;

    myRobot.setDestinationGPSPosition(5, 6);
    myRobot.setRobotGPSPosition(10000, -100000);

    int* destinationGPS = myRobot.getDestinationGPSPosition();
    int* robotGPS = myRobot.getRobotGPSPosition();

    cout << "Starting Destination x coord: " << destinationGPS[0] << endl;
    cout << "Starting Destination y coord: " << destinationGPS[1] << endl;

    cout << "\nStarting Robot x coord: " << robotGPS[0] << endl;
    cout << "Starting Robot y coord: " << robotGPS[1] << endl;

    myRobot.move();

    cout << "\nFinal Robot x coord: " << robotGPS[0] << endl;
    cout << "Final Robot y coord: " << robotGPS[1] << endl;

    return 0;
}
```

Construction: Test

Test case	001 – Movement class
Brief description	Tests whether the robot will successfully get to the destination. Negative input, positive input, and large numbers will be tested
Expected output	At the end of the test, robot location shall be the same as the destination location
pass or fail	pass
Date	6.25.2020

Comments	No errors found. Tested only integers. Need to test for floats in the future

```
Starting Destination x coord: 5
Starting Destination y coord: 6

Starting Robot x coord: 0
Starting Robot y coord: 0

Robot is walking towards location...

Final Robot x coord: 5
Final Robot y coord: 6
```

```
Starting Destination x coord: 5
Starting Destination y coord: 6

Starting Robot x coord: -10
Starting Robot y coord: -10

Robot is walking towards location...

Final Robot x coord: 5
Final Robot y coord: 6
```

```
Starting Destination x coord: 5
Starting Destination y coord: 6

Starting Robot x coord: 10000
Starting Robot y coord: 100000

Robot is walking towards location...

Final Robot x coord: 5
Final Robot y coord: 6
```

```
Starting Destination x coord: 5
Starting Destination y coord: 6

Starting Robot x coord: -10000
Starting Robot y coord: 100000

Robot is walking towards location...

Final Robot x coord: 5
Final Robot y coord: 6
```

```
Starting Destination x coord: 5
Starting Destination y coord: 6

Starting Robot x coord: 10000
Starting Robot y coord: -100000

Robot is walking towards location...

Final Robot x coord: 5
Final Robot y coord: 6
```

```
Starting Destination x coord: 5
Starting Destination y coord: 6

Starting Robot x coord: 5
Starting Robot y coord: 6

Final Robot x coord: 5
Final Robot y coord: 6
```

Formal technical review:

The formal technical review will be done in a setting where all members attending the meeting are familiar with the code. Then, the person who produced the code must give an agenda, a brief introduction, and finally a walk through of the code.

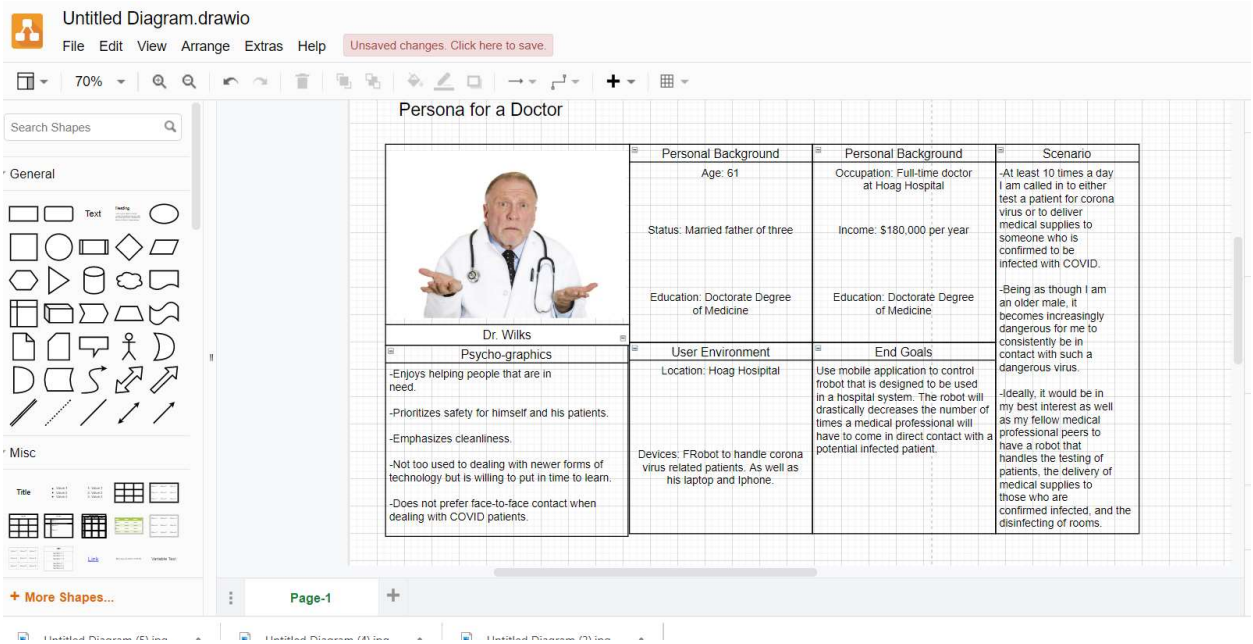
Naoki will present the code to Garinn who will raise any concerns shall he find any. This meeting will be recorded with a video camera so that Garinn can later document the meeting. Garinn will also need a full day in advance to review the code that Naoki has produced. At the end of the review, Garinn will give a judgement as to whether 1. Accept the product without modification or 2. Reject the product or 3. Accept with modifications. After the review, there will be a sign off between both Naoki and Garinn.

CASE Tools

Draw.io is the computer aided software engineering (CASE) tool that our team chose to use. In our case, Draw.io was used to ensure a high-quality and defect-free project design. This CASE tool ensured a check-pointed and disciplined approach and helped us as designers to see the project milestones during development. Draw.io illustrates a wide set of labor-saving tools that are used in software development. It generated a framework for organizing diagrams and to be helpful in enhancing productivity. One of the major advantages of using Draw.io is the delivery of the final product, which is more likely to meet real-world requirements. Here are a few more advantages that we found Draw.io to be helpful in using:

- As special emphasis is placed on redesign as well as testing, the servicing cost of a product over its expected lifetime is considerably reduced.
- The overall quality of the product is improved as an organized approach is undertaken during the process of development.
- CASE indirectly provides an organization with a competitive advantage by helping ensure the development of high-quality products.

An example of how we used our CASE tool, Draw.io, can be demonstrated in the screen shot of our Persona diagram which can be seen below:



We used Draw.io for the following segments of the project:

- (Analysis) Class Diagram (ACD)
- Class-Responsibility-Collaborator (CRC) cards
- Activity Diagram (AD)
- Swim-Lane Diagram (SLD)
- State Diagram (SD)
- (Analysis) Sequence Diagram (ASD)
- (Design) Class Diagram that show all design classes
- Architectural Structure Diagram
- State Chart (State Transition Diagram)
- Persona (user modeling)
- Preliminary Screen Layout
- Diagram for mapping user objectives into the interface structure
- MVC Architecture

Deployment

No activity for deployment, deliver, support, and feedback is required in this HW

References

None

Team Charter

Course Title	CPSC 362 Foundations of Software Engineering	All team members participated in the creation of this charter and agree with its content. Date 05/28/2020
Instructor	Dr. Chang-Hyun Jo	
Course Dates	05/26/2020 – 06/25/2020	

Team Members (Contact Information)

Name	Address (city, state, country)	Phone	Cell	Email

Garinn Morton	Costa Mesa, CA, USA	N/A	949-579-0297	gmorton2@csu.fullerton.edu
Naoki Atkins	Foothill Ranch, CA, USA	949-378-9543	949-3789543	naokishami@csu.fullerton.edu

Team Member Skill Inventory (Areas individual members can contribute)

Garinn Morton	▪ MS Word, Draw.io, Linux ▪ Programming Languages: C++, C, Java, C#, Assembly, PHP
Naoki Atkins	▪ MS Word ▪ Programming Languages: C++, Java, PHP, Javascript

Team Goals (Project goals, team process goals, quality goals, etc.)

- Learn about software processes and software design and architecture.
- Develop a strong, cohesive team and collectively produce the required assignments in a timely fashion.
- Acquire practical software engineering (people, process, tools, and methods) knowledge from teammates.
- Maintain great relationship between teammates.
- Produce and deliver a good final paper to the professor.
- Develop skills to facilitate future career goals.

Team Roles (Define roles of each member to achieve goals)

Garinn Morton	▪ Develop different graphs ▪ Aid in documenting the project
Naoki Atkins	▪ Help team members ▪ Document the project

Ground Rules (Meeting schedule/locations, attendance expectations, agenda, assignment completion, communication methods, etc.)

- Assist team members when they need help
- Be punctual to meetings
- Be cooperative
- Participate in the Discord chat

Time Commitments/Availability (Pacific Time)

Garinn Morton	▪ 12pm – 8pm Mon - Sat
Naoki Atkins	▪ 10am – 8pm Sat - Sun

Conflict Management (What are potential conflicts that might arise among or between team members during this course? How will team members deal with these and other conflicts?)

- If conflicts arise, bring it up to the Discord chat

Risk Management (What are potential barriers to the achievement of these goals?)

- If there's noticeable delay in developing an artifact, move on to the next artifact
- Always look back to the FR's to ensure that we're incorporating all the requirements

Team Evaluation Criteria (List evaluation criteria that will be used to evaluate team members objectively.)

- Number of artifacts produced
- Cooperativeness
- Responsiveness

Team Evaluation – HW1

(Naoki's Evaluation)

Garinn Morton: 100%

Naoki Atkins: 100%

(Garinn's Evaluation)

Garinn Morton: 100%

Naoki Atkins: 100%

Team Evaluation – HW2

Members / **Evaluators**	Garinn Morton	Naoki Atkins	**Total**	**Comments on Your Evaluation on Team**
Garinn Morton	100	100	200	All members did well.
Naoki Atkins	100	100	200	All members did well
GarinnMorton	100	100	200	
Naoki Atkins	100	100	200	
Total	400	400	800	
Max	600	600	600	
Average	100.00	100.00	200.00	
Percent	66.67%	66.67%	200.00%	
Signature				
Comments on Your Score Earned from Team	I deserve my score.	I deserve my score.		