CPEN 291

Project 2 Report

A. Group Info, project title and contribution summary

Group #: ___G6-B_____ (*Example: G1-L2B*)

Kaiwen Deng	Linxin Li
Shangzhou Xia	Ruiqi Tian

Student names:

Project title: Gotham's Delivery Service

Contribution summary:

Shangzhou(Simon) Xia: 1. Ordering page: HTML, CSS, JS; 2. Status Page: HTML, CSS, AJAX. 3. Login page: HTML,CSS,JS. 4. NodeJS: creating server and displaying files.

Kaiwen Deng is mainly working for HTML and CSS of the web. He is also responsible for some of the animation of the website using javascript.

Ruiqi Tian is responsible for back-end programming to update web pages and send signals to the robot. He is also in charge of setting up the virtual machine and making gif animations.

Linxin Li (Hizan Li) worked on:

1. Design, build, and improve the robot using parts from LEGO.
2. Design, build, and improve the map on the foam board.
3. Create a python server program using TCP to communicate between server and robot, details are described in part E and F.

B. Introduction and description

Online shopping has become prevalent in this era. This project emulates the online shopping user experience as well as a small scale reality represented by lego car and foamboard map.

Users can play with the login and ordering page, which are purely frontend-based.

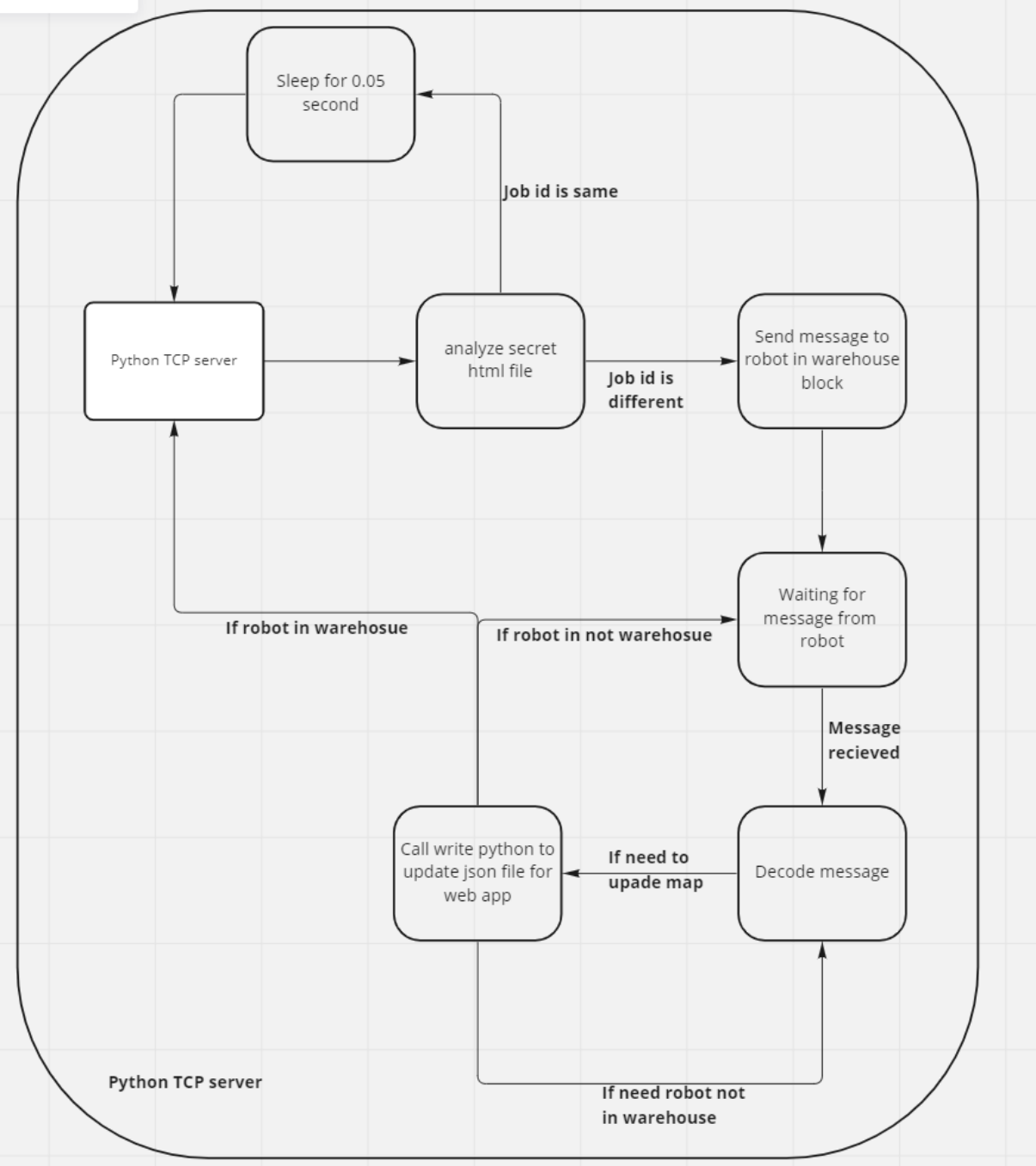
The main features are the status and map page, in which users can view where the lego car is on the map. For status, users can see a table specifying where this package is delivering to, how long it takes to arrive and what stage it is on (eg. the car just set off), and it has a progress bar showing different percentages of the completion. For the map, the tracking gif is displayed based on where the car is.

We think it's very cool! We want to build an attractive and extraordinary user interface of an online shopping web. The trade offs include time to demo, size of life lab room vs complexity of the map.

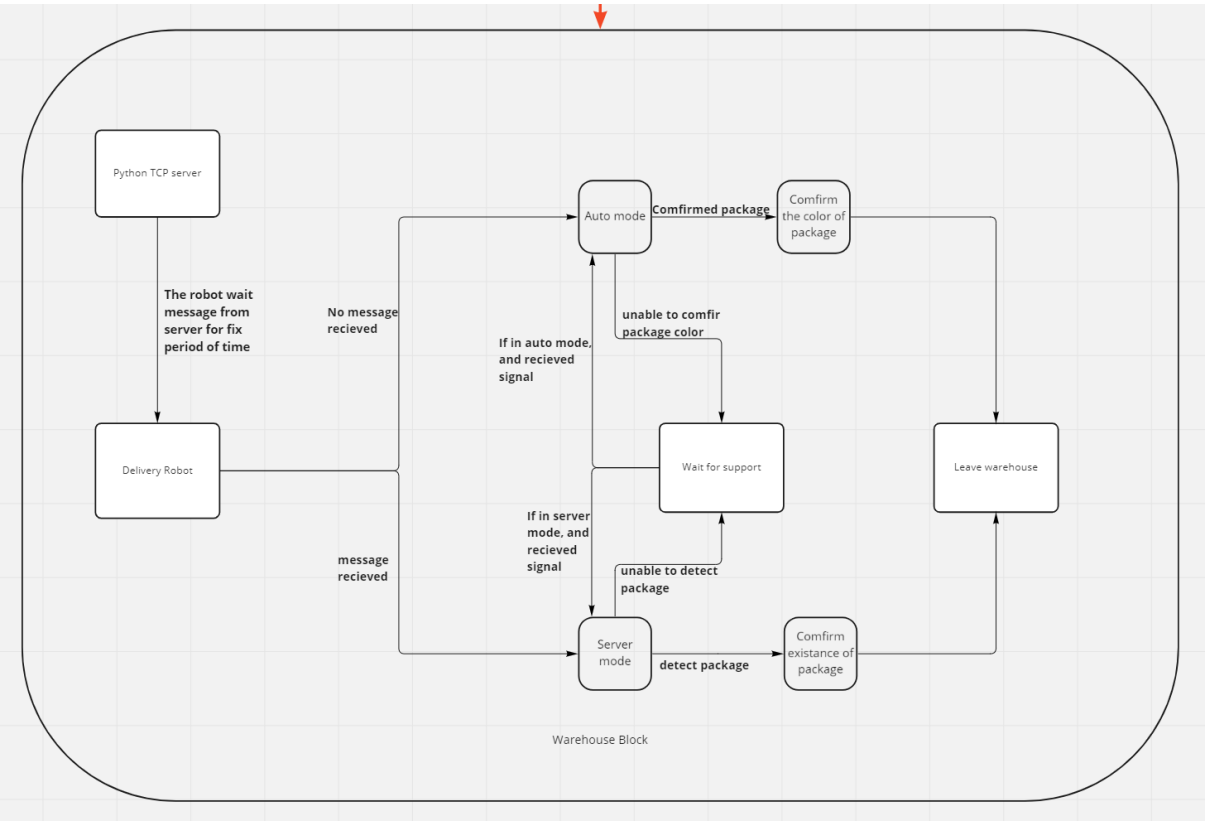
We first just want to build a lego car that's similar to a mario kart. Then we want to add more features to it, like carrying items. Then maybe a scenario may be more cool to display it. So we decided it can run on a designed map. Lastly, we use a web to emulate online shopping like Uber Eat/Amazon.

C. Block diagrams

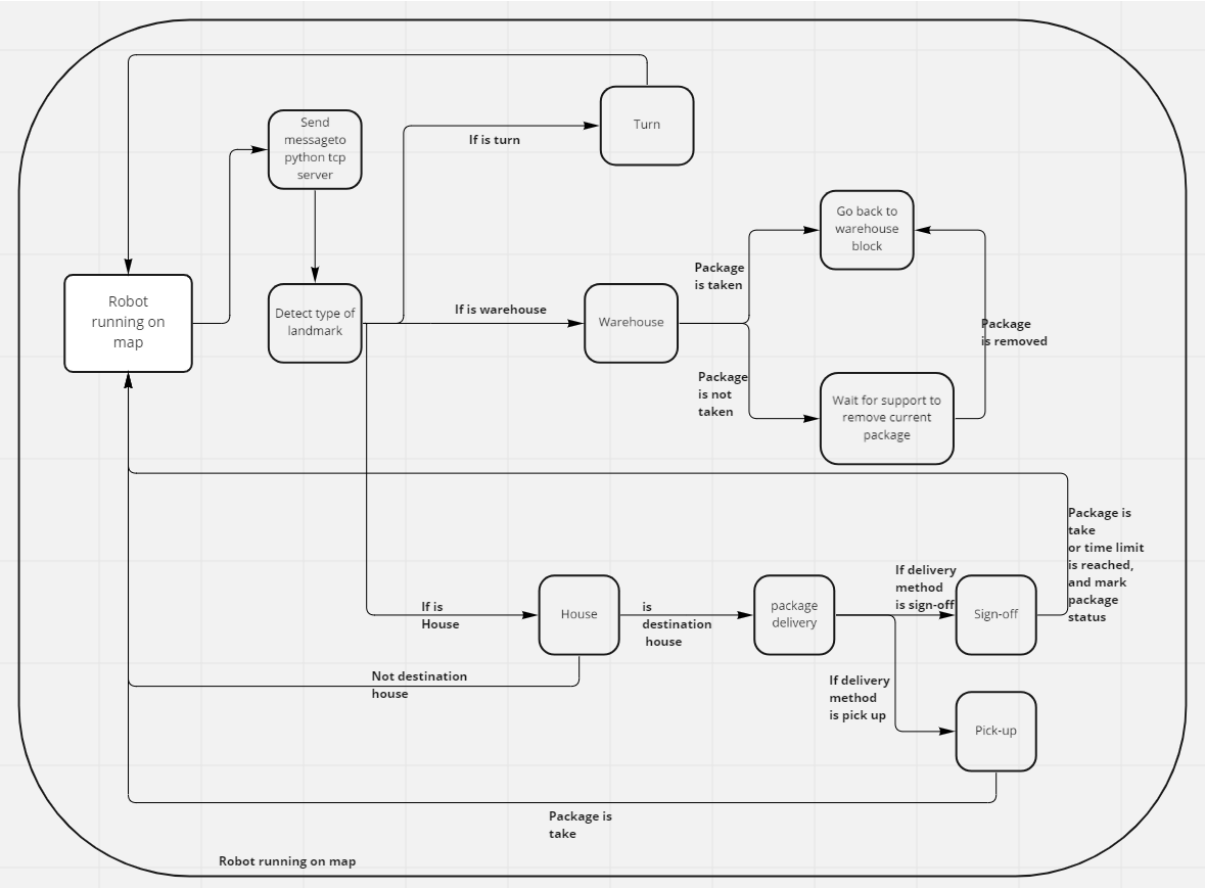
Block Diagram for Python TCP server:



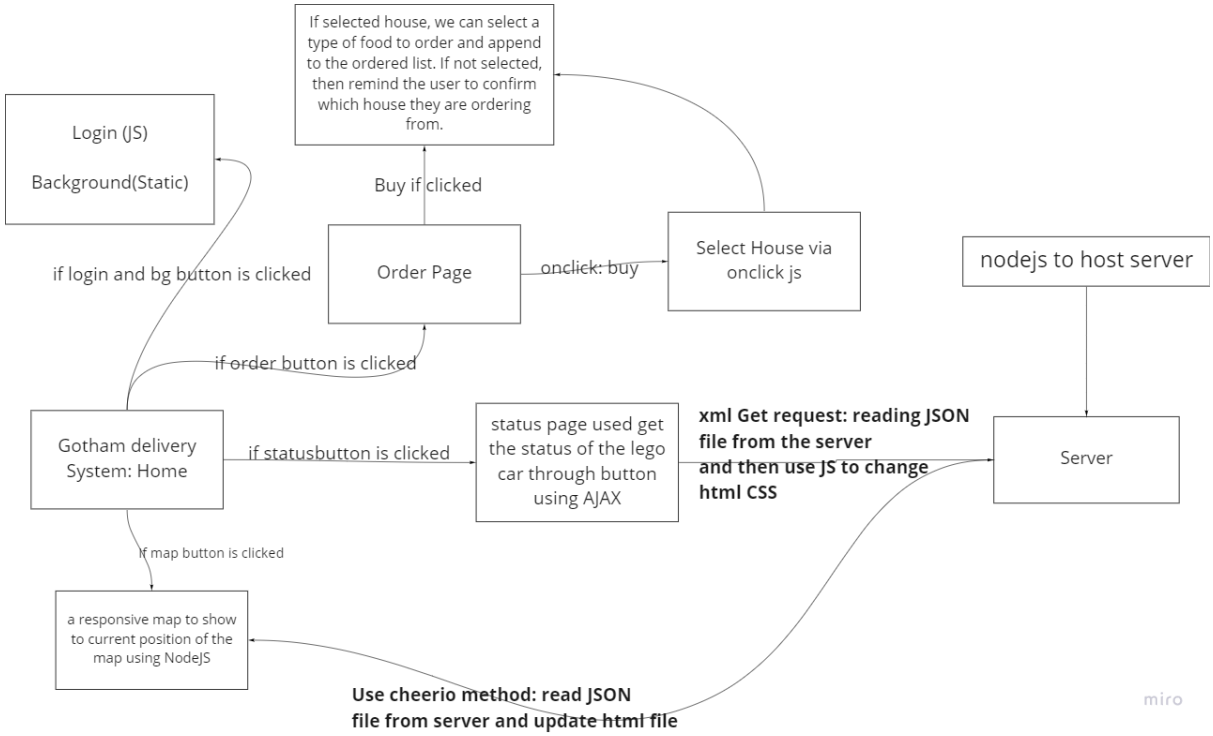
Block diagram for robot in warehouse block:



Block diagram for robot in running on map:



Block diagram of software:



In Sections D, E, and F of the report:

.

D. Technical documentation for the software [web app]

What technologies are used for the front-end:

-HTML, CSS, JS, BOOTSTRAP, AJAX

What technologies are used for the back-end:

-NodeJS, XML, AJAX, JSON

The front-end design and implementation

In this project, we decided to make a website that is similar to Amazon/Uber eat that can provide users the functionalities to order, track the status as well as seeing the map of the delivery car. For the front end, we used an extensive amount of HTML, CSS and JavaScript. We made several websites and allowed users to go to the webpage the user wants. For example, if the user wants to order, we have a button in the middle of every webpage because this is our main function of the webpage. We also have a status webpage to check where the lego car is and when it will possibly arrive. We also provide a map to allow visually seeing the car location. Beside that, we decorated our website using CSS/BS5 and made it more appealing to the user. We add animation to features such as buttons and images to make it react to the hover effect. Beside that, we have made a login page which allows the user to “login”(purely frontend though). However, due to the time constraint, we have not included SQL and database and thus only have a web page for login but not actualize its function.

The back-end design and implementation

For the backend part, we first used Apache and then switched to node.JS because we think node js can provide us with more functionality. We created a server using node js and using that, we can display our website on the server. To enable communication between the server and the lego car, we use a json file. Whenever the lego car moves to a landmark position, the lego car will send an updated json file to the server and the STATUS and MAP pages will get the update of the server and show it on the website.

E. Technical documentation for the software [excluding the web app]

Our web app will generate a secret html file when processing a new order from the website. For the python TCP based server program (not the web app, will be call PTS in content below), our PTS will read the secret html file generated by our web app, this secret html will contain three fields:

- First filed is the destination of this package;
- Second filed is the delivery method of this package (drop off or signed off)
- Third field is the jobid of this package

Then, our PTS will read this secret html file, and if the jobid is different from previous reading, the server will send a message in a certain format to the robot. Then the server will wait for messages from the robot.

- First character represents the destination of this package;
- Second filed is the delivery method of this package (drop off or signed off)

Robot will send its location and package status to the server when lopping on our map. After receiving a message from the robot, TCP will call another program, write.py to change a json file that controls the layout of our web page.

Table of all libraries used:

Library Name	Description	In which file
sys	Write to a file depend on the input received from robot	write.py
os	We can let the python program to automatically generate/modify file depend on the data sent by the robot	echo_server.py
socket	Using TCP to communicate between server(our 291 server) and client(robot)	echo_server.py

F. Technical documentation for the hardware/IoT

The robot has two modes, one is auto mode, other is server mode.

In server mode, the robot will wait for a period of time to get a message from the server, if the message is received, the robot goes to server mode, else the robot will go to auto mode. In this mode, the robot will deliver the package based on the message received from the server, not the colour of the package.

In auto mode, the robot will detect the package colour automatedly. It uses a colour sensor and an algotherm to confirm the colour on the package, making sure that it will deliver to the correct destination. In this mode, the robot will deliver the package based on the colour of the package, not the message received from the server.

For both auto mode and server mode, the robot will **only** leave the warehouse when the package is **detected** (server mode) or **confirmed** (auto mode) on its robotic arm.

If the robot can not detect or confirm the package, it will report this situation by speaking out. And wait for the package to be loaded on its robotic arm.

After the package is successfully detected or confirmed, the robot will leave the warehouse. When running on the map, the robot will send its location to the server in a fixed period of time. The message contain three character:

- First character represents the package status,
- Second character represents the location of the robot
- Third character represents the package colour

When the robot arrives at the destination, if the delivery method is pick-up, the robot will wait for a fixed period of time to let the owner take the package. If the delivery method is sign-off, then it will wait forever until the package is taken.

When the robot is running in auto mode, the default delivery method is in pick up.

Before entering the warehouse, if the package is not taken, then the robot will speak out to let people know that it can not load a new package, and wait until the package is taken.

After the robot enters the warehouse, it will wait for a message from the server, if the message is received, the robot goes to server-mode, else goes to auto mode.

Table of libraries used:

Library Name	Description	In which file
`socket`	Used to communicate with server	ev3dev.py
`pybricks`	Used to control the behaviour motors and read value from sensors	ev3dev.py

Table of components used, more detail list please see:

Part #	Quantity used
LEGO® MINDSTORMS® EV3 Brick	1
LEGO® MINDSTORMS®Motor	3
LEGO® MINDSTORMS®Motor	1
LEGO® MINDSTORMS®Sensor	1
LEGO® MINDSTORMS®Sensor	1
LEGO® MINDSTORMS®Sensor	1
LEGO® MINDSTORMS®Sensor	1
AVerMedia WebCam	1
Rechargeable Battery	6
Wifi node	1

G. Test and evaluations

For frontend testing, we mostly use different browsers and screen size to display our content. If something does not look right or satisfying, which is majorly the issue of styling, then we change some css to make it work. So the testing was frequent and with many iterations.

For backend testing, we obtained information from the try catch error messages after running the node environment. We also try different order conditions to test the accuracy of our robot.

We were stuck at creating a server on VM for a long time and then later figured out the problems could be solved via port 80 and “sudo node”. We should have raised the issue to the professor or classmates earlier, so that we could save a lot of time. We also struggled to get the robot to communicate with the server, and lastly we figured out the robot could also host a server. Then we used socket and TCP protocols so that the robot can manage to communicate with the server and receive/send signals.

The robot sometimes encountered pivoting problems, and we adjusted turning angles and speed to fix the issues.

H. Conclusions and Reflections

In this project, the most interesting tricks that we learned is building connections between the server and local. Before this project, we only did webpages on the local host. However, when it needs to connect to the server, the difficulty increases a lot. It is not like doing it on the local host at all. You need to use the get and post method as well as writing a json file to connect with each other. This is a very important topic and it provides us with a solid base.

Another interesting concept is that lego can be programmed through python.

In addition, we learnt that time management is very important. During the project period, there are a lot of other projects to do and thus we have relatively slow progress during the first two weeks. As a result, we have to spend a lot of time on every part in the last week which is very stressful. We did relatively well on team work since we separate the work evenly and everyone has done their job perfectly. In terms of project management, I think we have a good plan but we are not able to achieve some of the functions. For example, we originally wanted to implement an order webpage that can directly transfer the order data to the car but eventually we did not finalise it. This shows that we probably need to follow our plan more strictly.

I. References and bibliography

Reference:

W3school.com

Stackoverflow.com

Nodejs.com

Files submitted on canvas:

File Name
order.js
order.html
order.css
mapfront.js
map.json
map.js
map.html
login.html
login.css
index.js
home.html
home.css
background.html
server.js
secret.html

Reference:

https://pybricks.com/ev3-micropython/index.html

Files submitted on canvas:

File name
ev3dev.py
echo_server.py
write.py

J. Answer the following questions

The following questions are related to the accreditation data collection related to graduate attributes.

Design Process for some core functionalities: Choose a different core component from the one(s) you have explained in the Milestone 2 report.

Describe clearly the process you have used for the following design aspects of some core components of your project (for example, IoT or Web, …).

Please spend time to discuss and carefully answer each of them.

1. **Use of process**: Describe your approach to adapt and apply a general design process for the core features in your project. What was your approach?

First, generating ideas. What our web and car do, and how they are related.

Second, identify our main functionalities.

Third, implement functionality via different methods.

Fourth, fail fast and adjust.

Fifth, iteration.

2. **Constraint identification**: Explain the constraints that you must consider in the design of those features/functionalities.

The hardware limits the amount of sensors to 8. And the software skills our group obtained.

3. **Solution generation**: Explain at least two possible alternative features that your group rejected due to technical reasons and explain why.

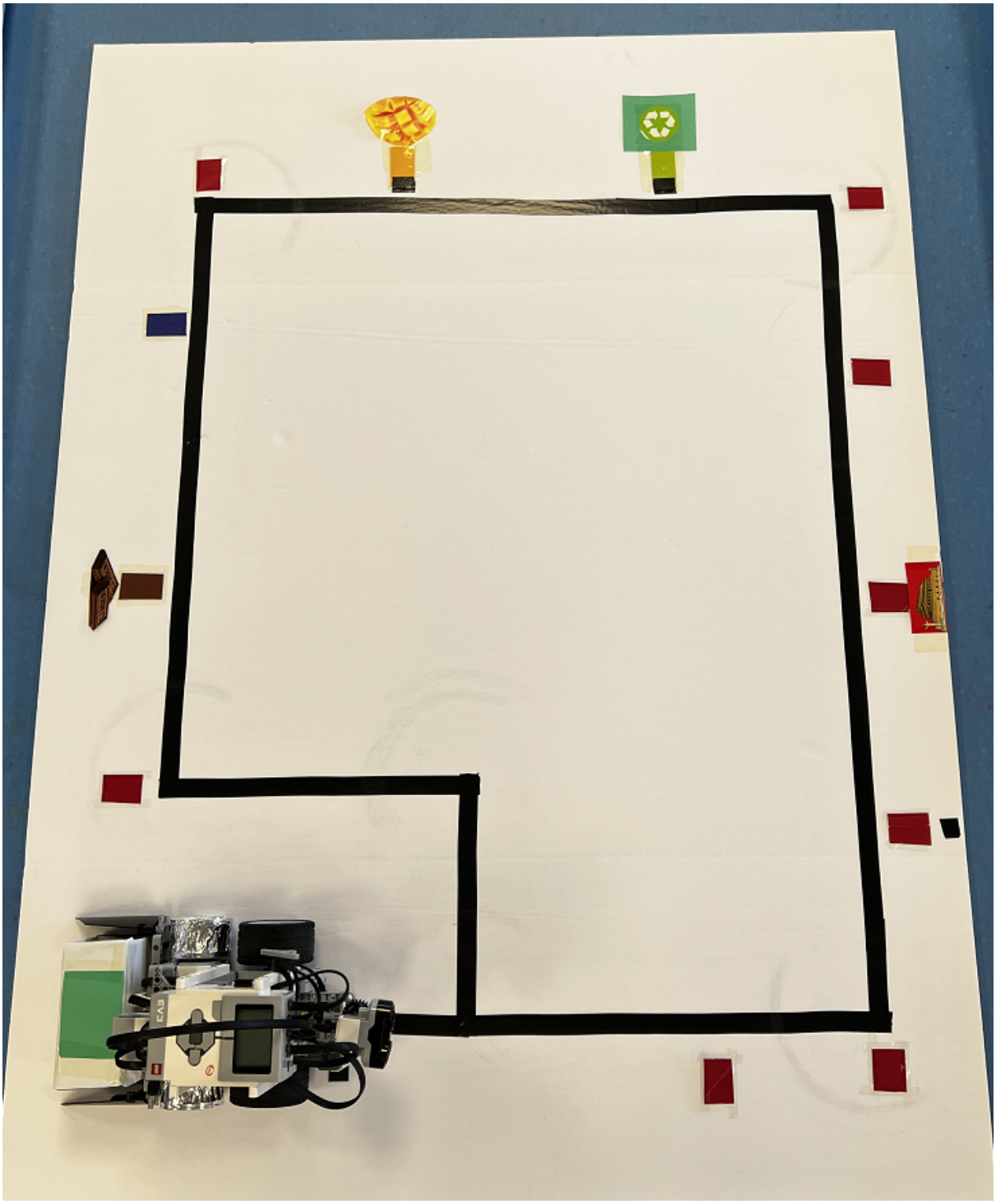
We could have used the fetch API or Axios to acquire JSON data but we use AJAXbecause of time constraints to learn fetch and use the express framework for axios, therefore we choose AJAX to implement the data getting job.

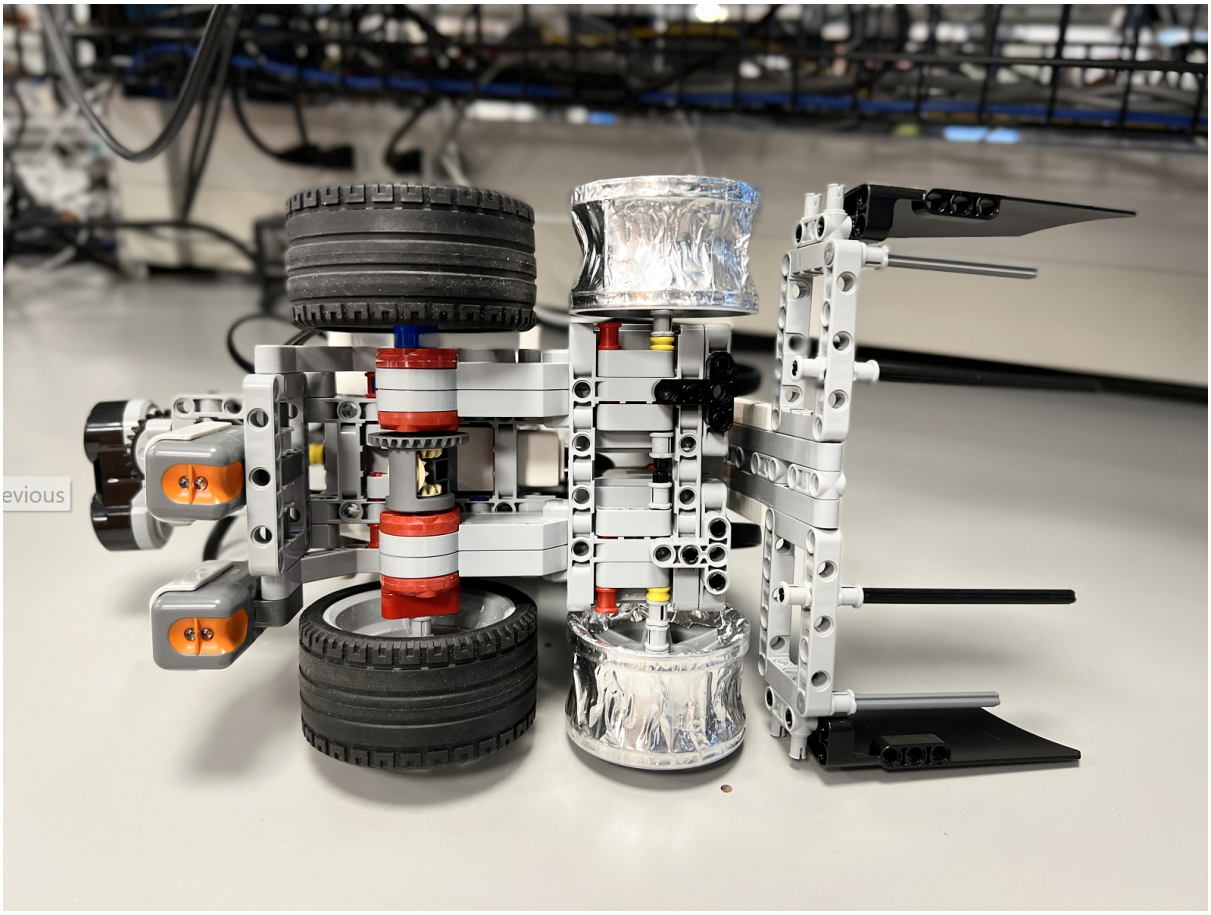
We could use a treadmill to roll down the package but we find it might damage the package in real life. Therefore, we use a “shover”.

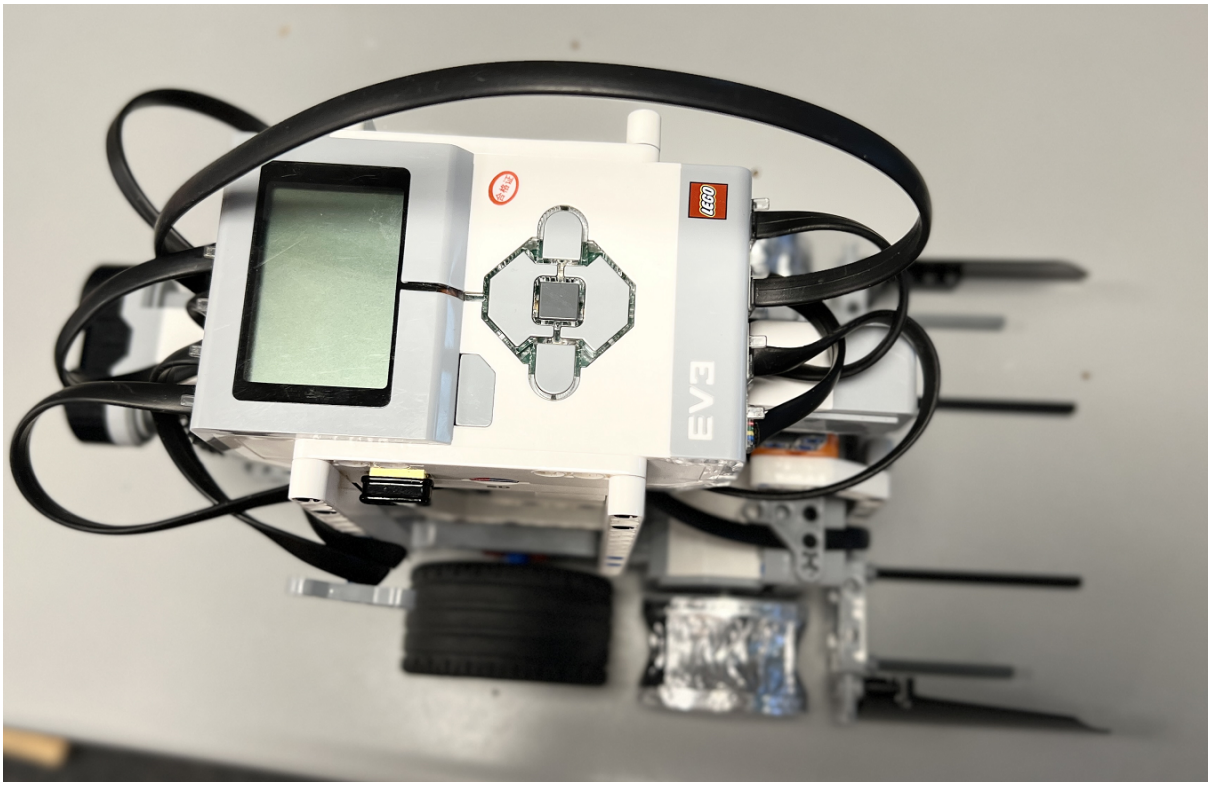
4. **Solution Assessment**: Explain how you tested and assessed the viability and then correctness of your group’s selected features.

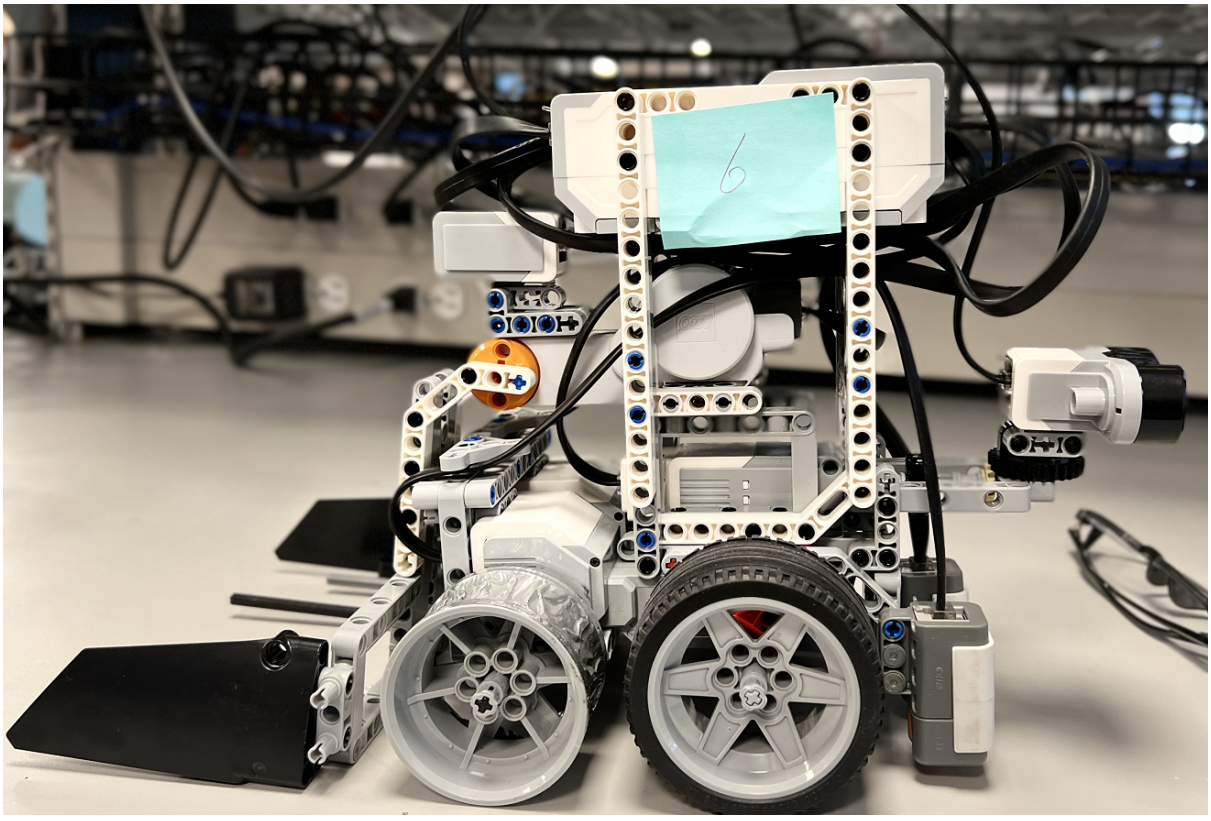
We ran the car on the map and tested different houses to ensure its correctness.

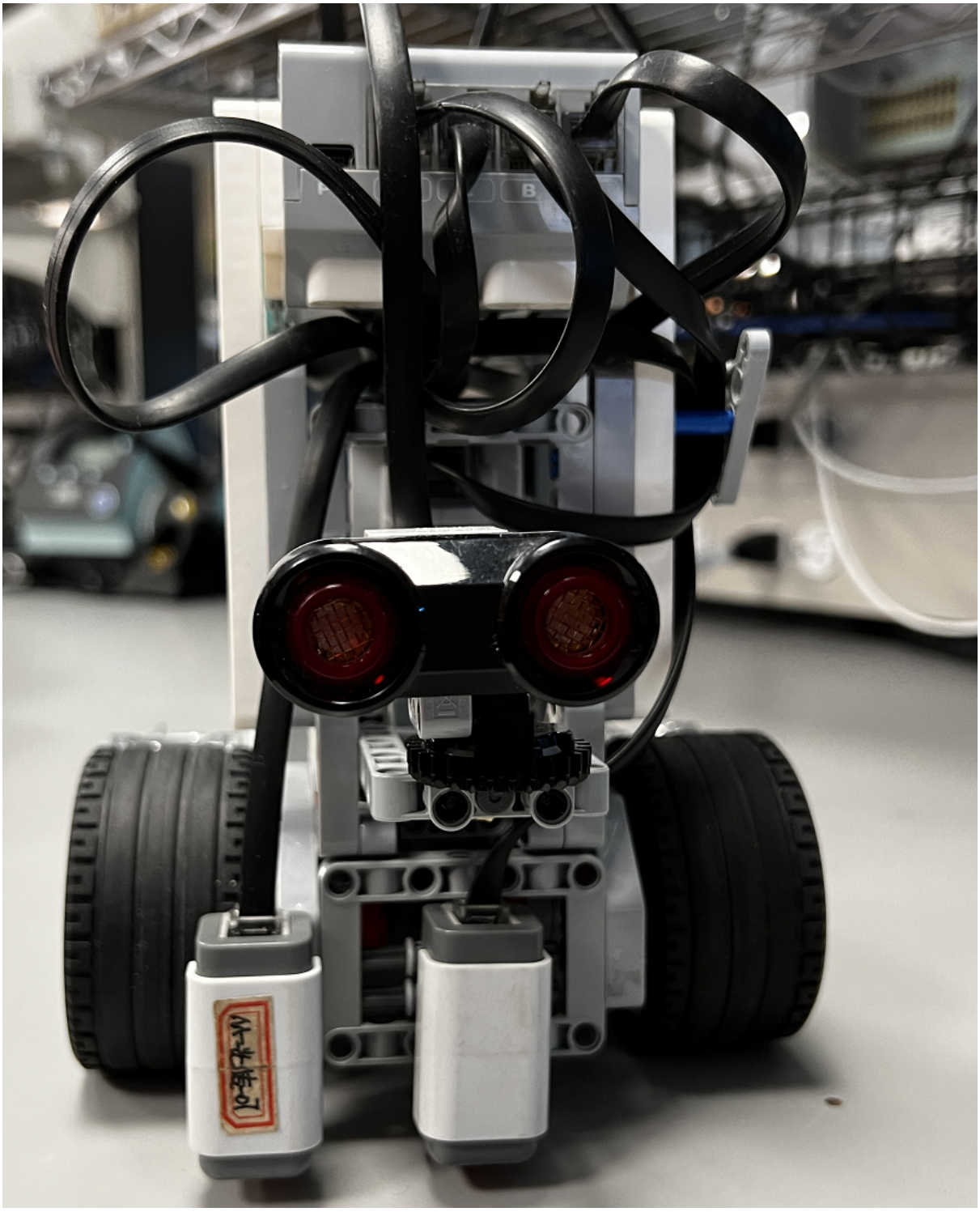
Appendix A – Project pictures (hardware)

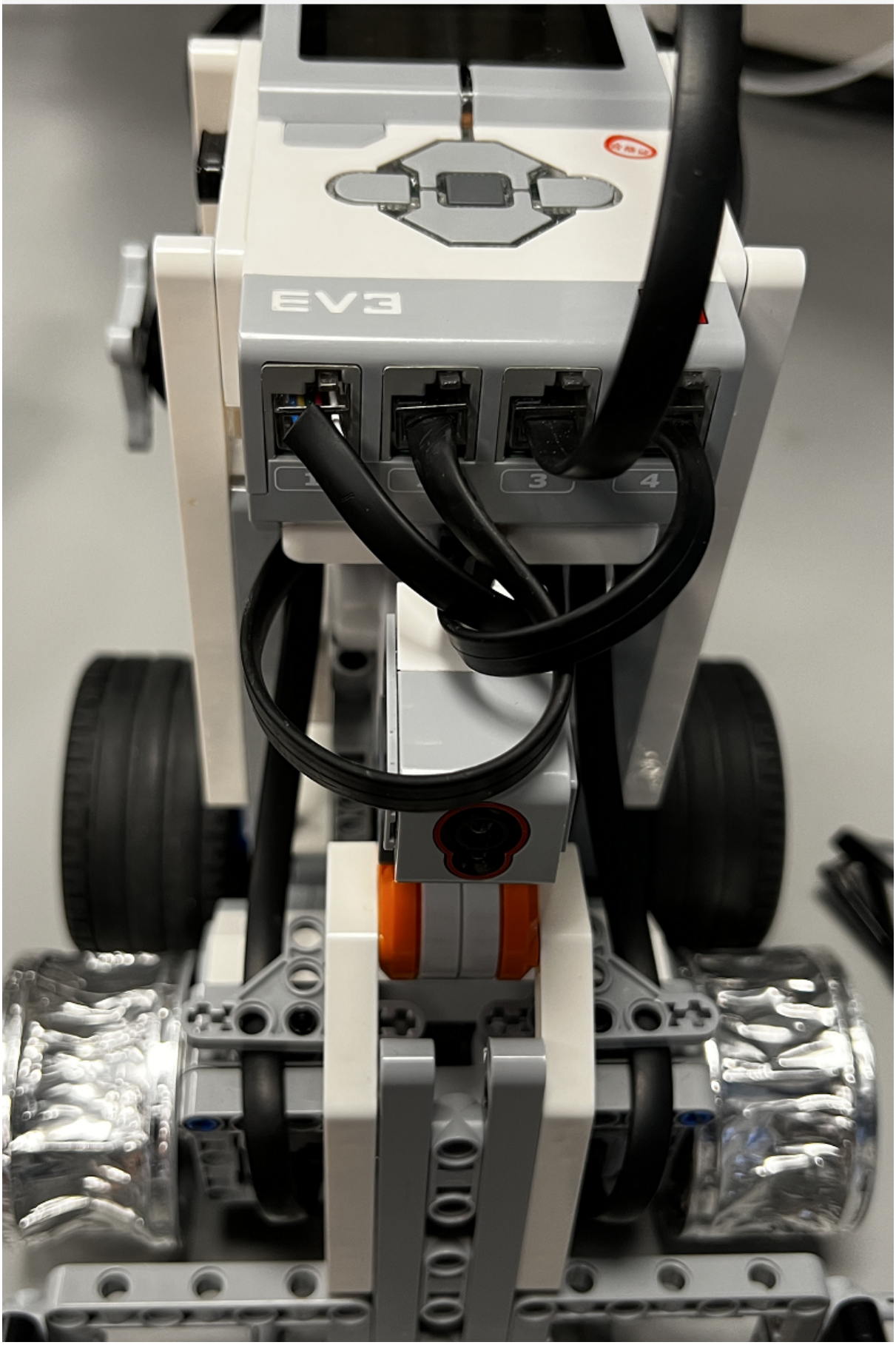





EV3

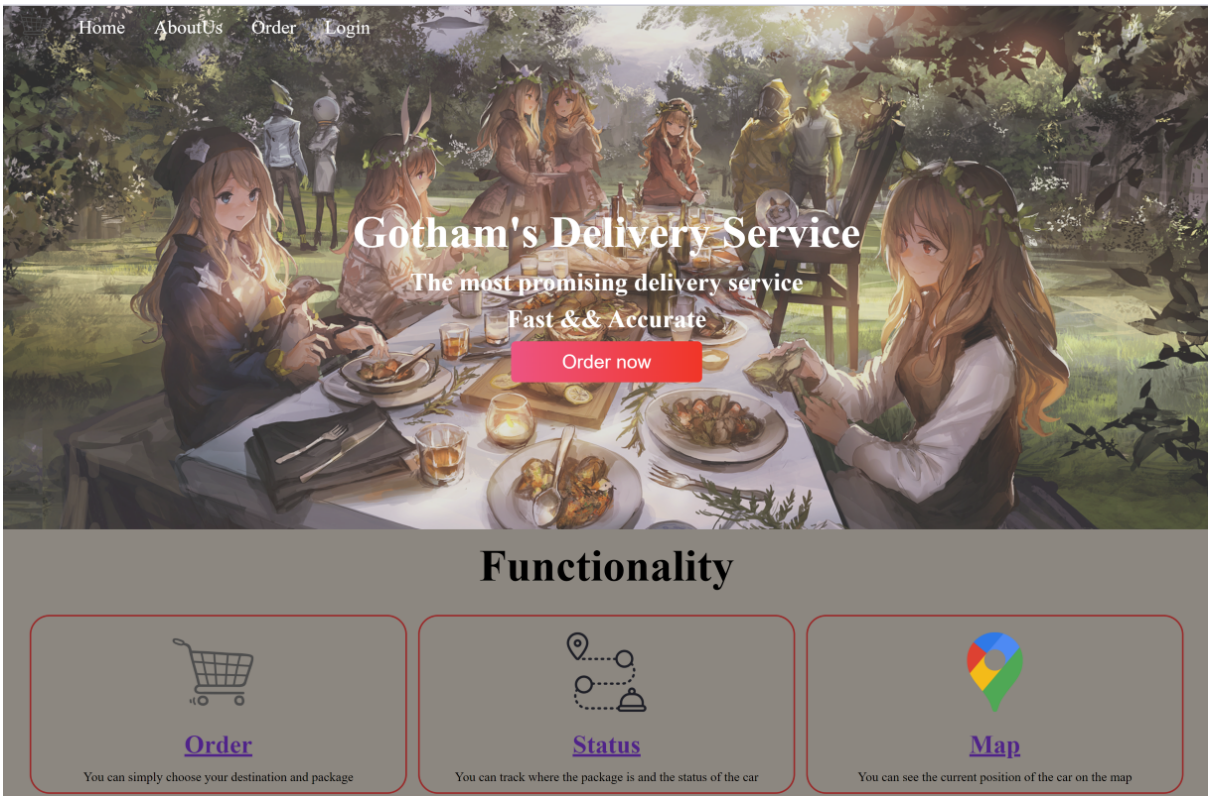

6

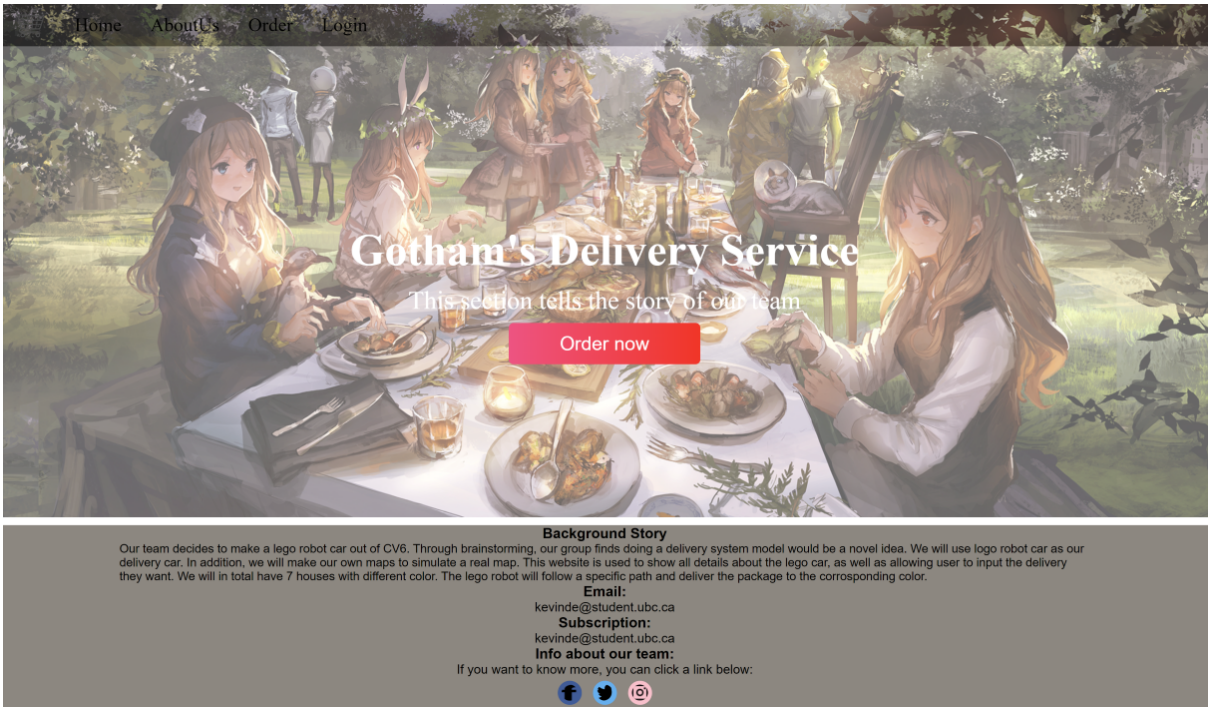


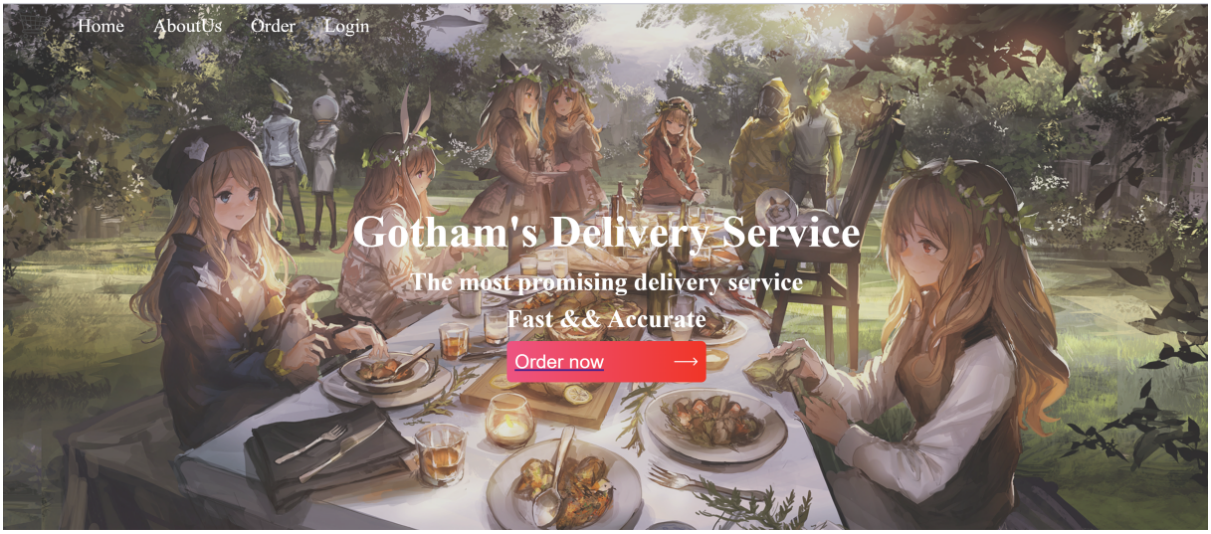

EV3
3
4

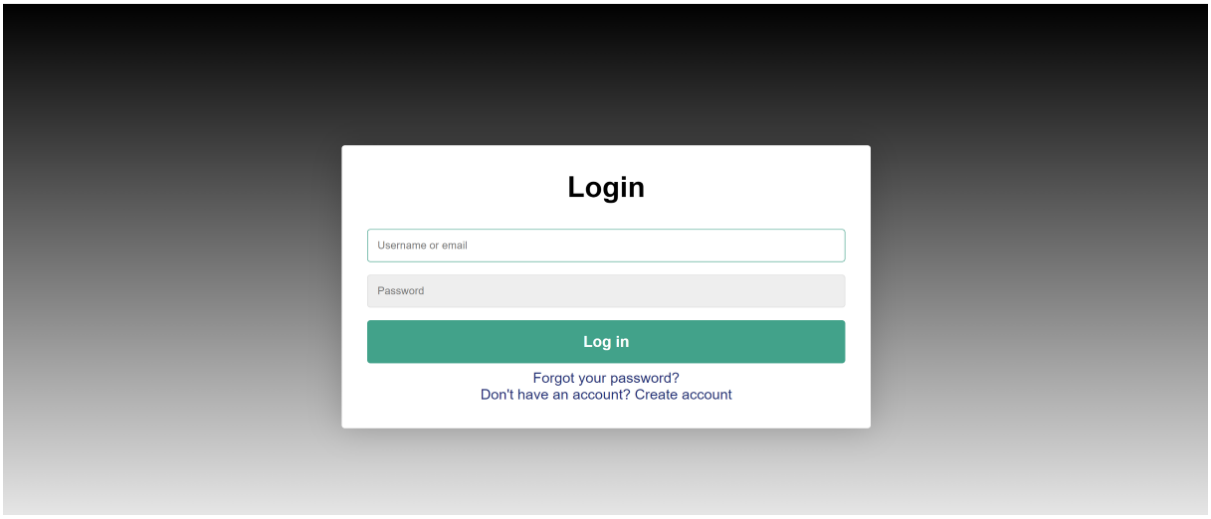
Appendix B – Project pictures (software)

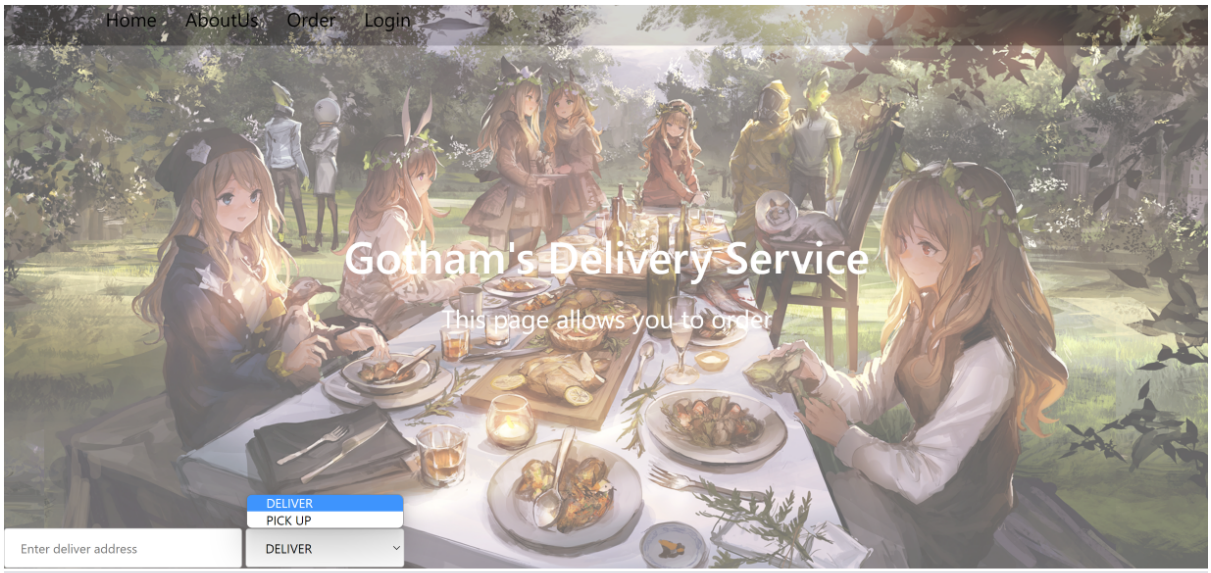
Include clear images/snapshots of the user-interface for software (Web app …).






Home
AboutUs
Order
Login
Gotham's Delivery Service
The most promising delivery service
Fast && Accurate
Order now


Login
Username or email
Password
Log in
Forgot your password?
Don't have an account? Create account



Dear customer, please click your house to make a order




House 1 | House 2 | House 3 | House 4

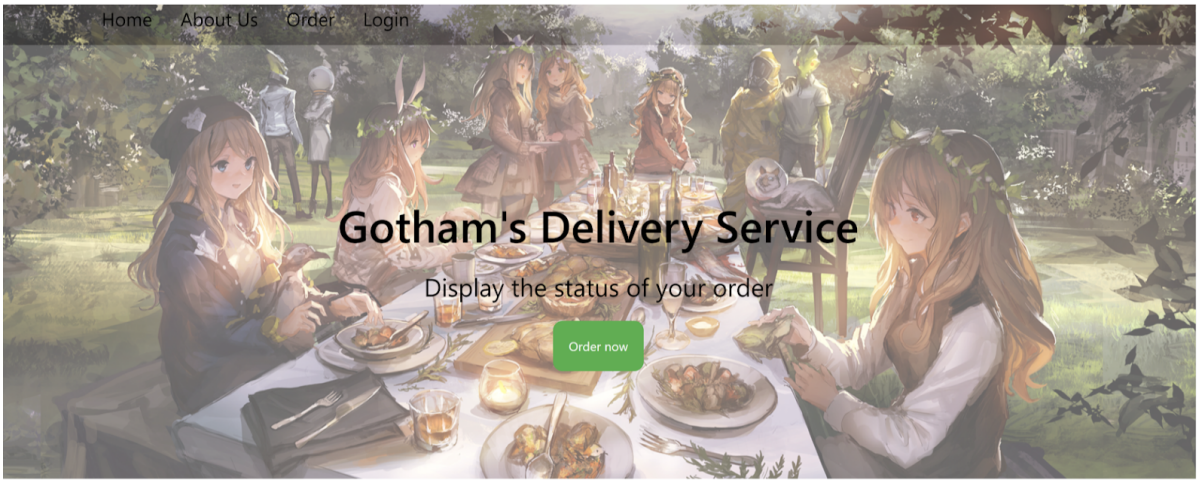
Different color corresponding to different color on the map

Nobody is shopping???

Steak

Guess your likes

What do u like to search? SEARCHHHH What do u like to order? Submit Remove



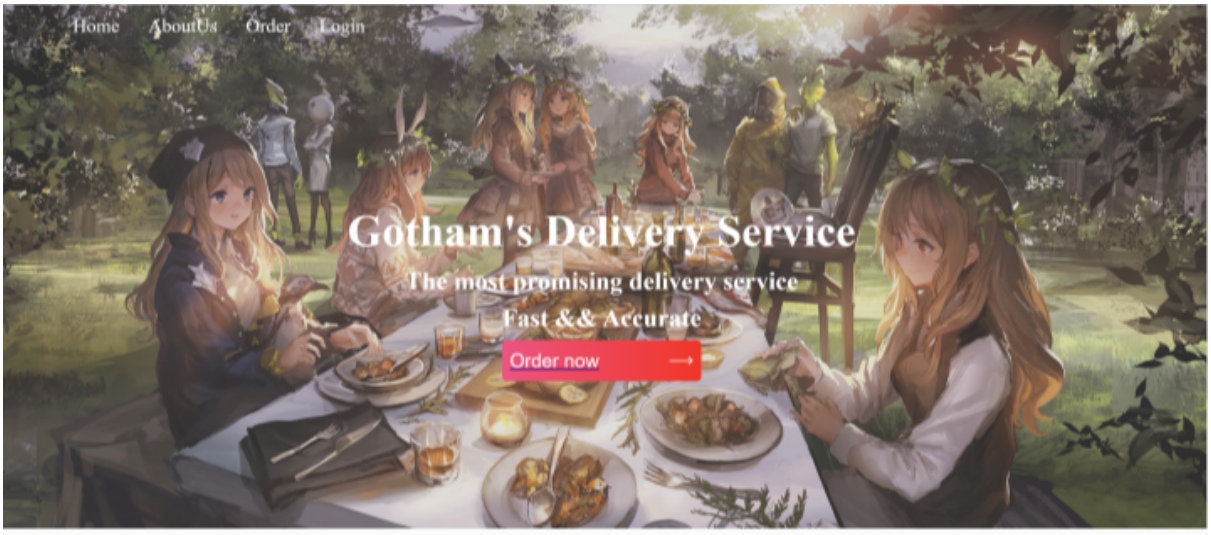
Location	Package Color	Estimated Delivery Time
Your delivery driver is on his/her way	Red	00:45

We are proud to deliver your package via the eletrical and sustainable EV3 LEGO CAR

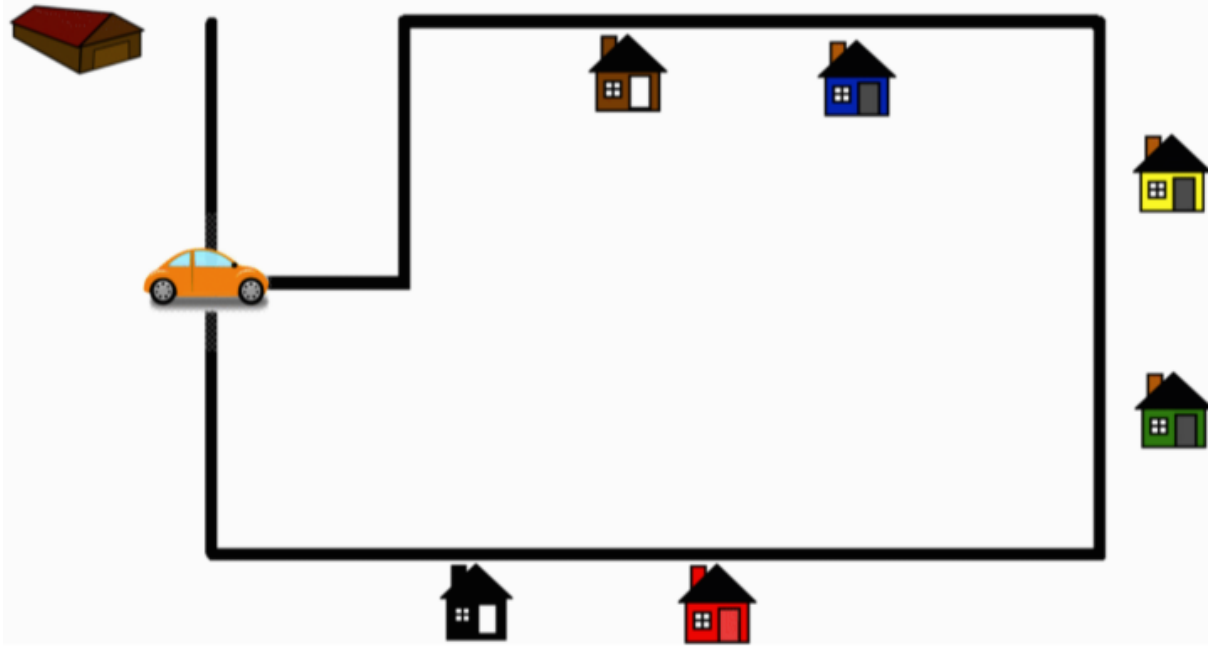
Basic Progress Bar

40% complete

update progress


Home
AboutUs
Order
Login
Gotham's Delivery Service
The most promising delivery service
Fast && Accurate
Order now

Working ...



Appendix C - Fritzing

We used ev3 lego. After communicating with professor Farshid, for the fritzing part, since we do not have any wires, we would just include the robot's detailed views and sensors.

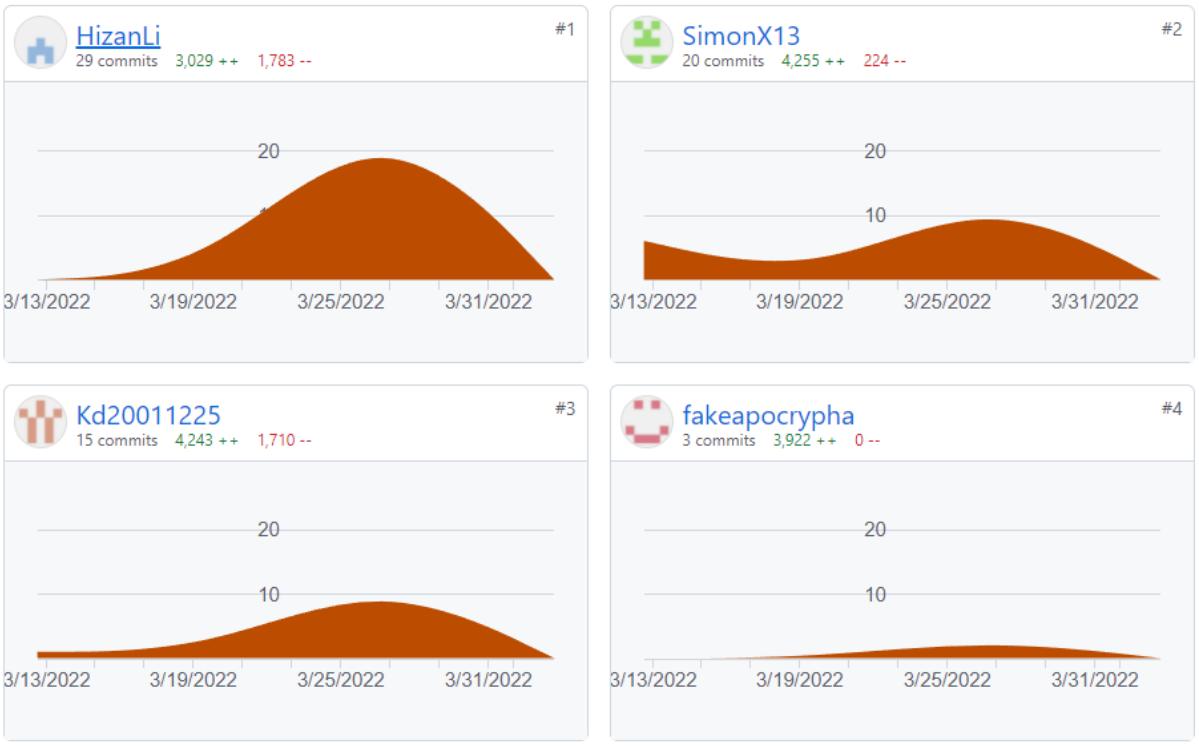
Lego EV3 Brick x1:

Ultrasonic sensor x1. Light sensor x3

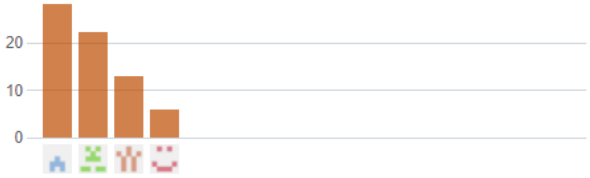
Medium size motor x1

Larger size motor x3

Appendix D - GitHub/Version Control



Excluding merges, **4 authors** have pushed **61 commits** to master and **69 commits** to all branches. On master, **114 files** have changed and there have been **11,056 additions** and **75 deletions**.



Appendix E – Component list

Part #	Description	Quantity used
LEGO® MINDSTORMS® EV3 Brick	Our microcontroller to control our robot, running linux and python, more detail get here https://pybricks.com/	1
LEGO® MINDSTORMS®Motor	One motor is used to pick up the package; Two motors are connected to the wheel, which control the motion of the robot. https://pybricks.com/ev3-micropython/ev3devices.html#motors	3
LEGO® MINDSTORMS®Motor	Used to control our Ultrasonic Sensor to scan around to avoid collision when moving	1
LEGO® MINDSTORMS®Sensor	Ultrasonic Sensor, used to detect obstacles	1
LEGO® MINDSTORMS®Sensor	Light Sensor, measures the reflection of a surface using a red light., used for line tracking	1
LEGO® MINDSTORMS®Sensor	Light Sensor, measures the reflection of a surface using a red light., detect landmark	1
LEGO® MINDSTORMS®Sensor	Color Sensor, used to detect and confirm the color of our package, and sent result to our microcontroller	1
AVerMedia WebCam	Used for track the location and motion of our robot	1
Rechargeable Battery	Used to power our robot	6
Wifi node	Used for EV3 brick to connect to WIFI and Bluetooth	1

Appendix F – Hardware-related Code

Code for ev3dev.py

```
import os
import sys
import socket
from pybricks.ev3devices import Motor, ColorSensor, UltrasonicSensor
from pybricks.nxtdevices import LightSensor
from pybricks.parameters import Port, Button, Color
from pybricks.tools import wait
from pybricks.robotics import DriveBase
from pybricks.hubs import EV3Brick

# possible colored_landmark on mpa
BLUE = 0
GREEN = 1
YELLOW = 2
TURN = 3
PURPLE = 4
BLACK = 5
WHITE = 6
RED = 7
BROWN = 8
AUTO = 9

#for turning
right = 1
left = 0

def start(right_motor,left_motor, arm_motor, ladar_motor, light_sensor_line,
light_sensot_landmark, color_sensor, ultrasonic_sensor):
    color_dict = {
        '0': BLUE,
        '1': GREEN,
        '2': YELLOW,
        '5': BLACK,
        '6': WHITE,
        '7': RED,
        '8': BROWN,
        '9': AUTO
    }
    me_dict = {
        "0" : 0,  # "pickup"
        "1" : 1   # "wait"
    }

    #read command from python ptc based server
    command = get_commmand()

    debug_print("command is "+ command)
```

```
    if command is None:
        auto_or_server = 0
        wait_or_pickup = 1
    else:
        result = color_dict[command[0]]
        wait_or_pickup = me_dict[command[1]]

    auto_or_server = 0
    if result in range(0,7):
        auto_or_server = 1

    debug_print(auto_or_server)
    debug_print(wait_or_pickup)


    turning_count = 0
    landmark_count = 0
    turning_direction = [left, left, left, left, right, left]

    landmark_order = [PURPLE,   # Ladmark for entering warehouse
                      TURN,     #check point 1 , and turn left
                      BLACK,    #Black House
                      RED,      #RED HOUSE
                      WHITE,    #WHIRE HOUSE
                      TURN,     #check point 2, and turn left
                      GREEN,    #green House
                      YELLOW,   #Yello House
                      TURN,     #check point 3, and turn left
                      BLUE,     #blue House
                      BROWN,    #brown House
                      TURN,     #check point 4, and turn left
                      TURN,     #check point 5, and turn right
                      TURN,     #check point 6, and turn left
                      ]

    #define our robot
    robot = DriveBase(left_motor, right_motor, wheel_diameter=70,
axle_track=104)

    ev3 = EV3Brick()
    ev3.light.on(Color.GREEN)
    ev3. speaker.set_speech_options(language='en', voice='f1', speed=110,
pitch=50)

    # Measured light threshold for different colour surfac.
    BLACK_reflection = 12
    WHITE_reflection = 95
    threshold = (BLACK_reflection + WHITE_reflection) / 2

    # Set the drive speed at 120 millimeters per second.
```

```
    DRIVE_SPEED = 120

    # Set the gain of the proportional line controller. This means that for
every
    # percentage point of light deviating from the threshold, we set the turn
    # rate of the drivebase to 1 degrees per second.
    # For example, if the light value deviates from the threshold by 10, the
robot
    # steers at 10*1 = 10 degrees per second.
    PROPORTIONAL_GAIN = 1

    back_to_warehouse = 0
    package_color = in_wareHouse(robot,
arm_motor,ladar_motor,light_sensot_landmark, color_sensor, ultrasonic_sensor,
ev3, auto_or_server, result)
    package_delivered = 0
    robot_location = 1

    # send the statue to our server (using tcp server client)
    send_status_to_server(str(package_delivered) + str(robot_location) +
str(package_color))

    while True:
        # Calculate the deviation from the threshold.
        deviation = light_sensor_line.reflection() - threshold
        # Calculate the turn rate.
        turn_rate = PROPORTIONAL_GAIN * deviation
        # Set the drive base speed and turn rate.
        robot.drive(DRIVE_SPEED, turn_rate)

        color = light_sensot_landmark.reflection()
        distance = ultrasonic_sensor.distance()

        # if a landmark is detected, then check which it block
        if(color < WHITE_reflection - 10 and color > BLACK_reflection + 30):
            landmark_color = landmark_order[landmark_count]
            landmark_count += 1
            landmark_count %= len(landmark_order)
            wait(250)

            if landmark_color is TURN:
                robot.stop()
                wait(250)
                turn = turning_direction[turning_count]
                turning_count += 1
                turning_count %= len(turning_direction)
                if turn is left:
                    robot.stop()
                    robot_location += 1
                    robot_location %= 6
                    if robot_location == 0:
```

```
                        robot_location = 1
                    # update statue to server
                    send_status_to_server(str(package_delivered) +
str(robot_location) + str(package_color))

                    ev3.speaker.say("Now turning left")
                    turn_left(robot, ultrasonic_sensor, ladar_motor, ev3)
                else:
                    ev3.speaker.say("Now turning right")
                    turn_right(robot, ultrasonic_sensor, ladar_motor, ev3)

            elif landmark_color is PURPLE:
                robot.stop()
                wait(250)
#if back_to_warehouse is one means we need to go to warehouse to deal with next
package
                if back_to_warehouse is 1:
                    if package_delivered is 1:
                        robot.straight(-400)
                        arm_motor.run_angle(100, 90)

                    else:
                        robot.straight(-200)
# package is not taken at delivery time
                        ev3.speaker.say("please remove package before load new
one!")
                        while True:
                            color = color_sensor.color()
                            if color is None:
                                break
                            wait(500)
                        robot.straight(-200)
                        arm_motor.run_angle(100, 90)

                    back_to_warehouse = 0
                    turning_count = 0
                    landmark_count = 0
                    package_delivered = 0

                    # change to auto mode
                    wait_or_pickup = 1
                    auto_or_server = 0
                    # update the status to server
                    send_status_to_server(str(package_delivered) + str(0) +
str(package_color))
                    command = get_commmand()
                    if command is None:
                        auto_or_server = 0
                        wait_or_pickup = 1
                    else:
                        result = color_dict[command[0]]
```

```
                        wait_or_pickup = me_dict[command[1]]

                    auto_or_server = 0
                    if result in range(0,7):
                        auto_or_server = 1
                    #
                    package_color = in_wareHouse(robot,
arm_motor,ladar_motor,light_sensot_landmark, color_sensor, ultrasonic_sensor,
ev3, auto_or_server, result)

            else:
                robot.stop()
                wait(250)
                #delivering package
                if landmark_color is package_color:
                    deliver_package(ev3, wait_or_pickup, color_sensor)
                    wait(250)
                    print("package_exist ")
                    # after deliver_package process, if color sensor can still
read date, it means package is not taken
                    if color_sensor.color() is None or
get_color_index(color_sensor.color()) is not package_color:
                        package_delivered = 1
                        send_status_to_server(str(package_delivered) +
str(robot_location) + str(package_color))
                    else:
                        package_delivered = 0
                    back_to_warehouse = 1


        if(distance < 50 ):
            robot.stop()
            ev3.light.on(Color.RED)
            while True:
                distance = ultrasonic_sensor.distance()
                ev3.speaker.beep(frequency=500, duration=100)
                if distance > 50:
                    ev3.light.on(Color.GREEN)
                    break
                wait(500)

# tell people to pick up their package
def deliver_package(ev3, wait_or_pickup, color_sensor):
    if wait_or_pickup is 1:
        ev3.speaker.say("please pick up your package in 5 count")
        count = ["five", "four", "three", "two", "one"]
        for i in count:
            ev3.speaker.say(i)
            if color_sensor.color() is None:
                break
    else:
```

```
        while True:
            color = color_sensor.color()
            if color is None:
                break
            ev3.speaker.say("please pick up your package")
        ev3.speaker.say("package is taken, leaving now")

# let robot turn left 90 degree
def turn_left(drive_base, ultrasonic_sensor, ladar_motor, ev3):
    drive_base.straight(5)
    check_safety(drive_base, ladar_motor, ultrasonic_sensor, ev3)
    drive_base.turn(-95)
    pass

# let robot turn right 90 degree
def turn_right(drive_base, ultrasonic_sensor, ladar_motor, ev3):
    drive_base.straight(30)
    check_safety(drive_base, ladar_motor, ultrasonic_sensor, ev3)
    drive_base.turn(95)
    pass


# the robot using ultrasonic_sensor to check it's sorrounding enviroment
def check_safety(drive_base, ladar_motor,ultrasonic_sensor, ev3):
    drive_base.stop()
    ladar_motor.run_angle(100, 210)
    for i in range(0,43):
        ladar_motor.run_angle(100, -10)
        distance = ultrasonic_sensor.distance()
        # if less than 5 cm, then stop
        if(distance < 50 ):
            ev3.light.on(Color.RED)
            while True:
                distance = ultrasonic_sensor.distance()
                ev3.speaker.say("obstacle detected, waiting for clear path")
                if distance > 50:
                    ev3.light.on(Color.GREEN)
                    break
                wait(500)
    ladar_motor.run_angle(100, 210)

# defines how robot behave when in wareHouse
def in_wareHouse(drive_base, arm_motor,ladar_motor,light_sensot_landmark,
color_sensor, ultrasonic_sensor,ev3, auto_or_server, stripped):
    drive_base.stop()
    ev3.light.on(Color.YELLOW)
    wait(200)
    arm_motor.run_angle(50, -90)
    wait(200)
    package_color = None
    if auto_or_server == 0:
```

```
        while package_color is None:
            package_color = detecte_package_color(drive_base,
arm_motor,ladar_motor,light_sensot_landmark, color_sensor,
ultrasonic_sensor,ev3, 1)
            debug_print("package_color is "+ str(package_color))
            wait(200)
            if package_color is not None:
                break
            wait(1000)
    else:
        detecte_package_color(drive_base,
arm_motor,ladar_motor,light_sensot_landmark, color_sensor,
ultrasonic_sensor,ev3, 0)
        if stripped is BLACK:
            ev3.speaker.say("recieved from server, now deliver to black house")
            package_color = BLACK

        elif stripped is BLUE:
            ev3.speaker.say("recieved from server, now deliver to blue house")
            package_color = BLUE

        elif stripped is GREEN:
            ev3.speaker.say("recieved from server, now deliver to green house")
            package_color = GREEN

        elif stripped is YELLOW:
            ev3.speaker.say("recieved from server, now deliver to yellow house")
            package_color = YELLOW

        elif stripped is RED:
            ev3.speaker.say("recieved from server, now deliver to red house")
            package_color = RED

        elif stripped is WHITE:
            ev3.speaker.say("recieved from server, now deliver to white house")
            package_color = WHITE

        elif stripped is BROWN:
            ev3.speaker.say("recieved from server, now deliver to brown house")
            package_color = BROWN



    ev3.light.on(Color.GREEN)
    check_safety(drive_base, ladar_motor,ultrasonic_sensor, ev3)
    drive_base.straight(200)
    return package_color

#this function check and confirm the color of our package, and decided which
house our robot will goes to
def detecte_package_color(drive_base,
```

```
arm_motor,ladar_motor,light_sensot_landmark, color_sensor,
ultrasonic_sensor,ev3, say):
    #check if there is package
    package_color = None
    while True:
        #first time reading color
        package_color1 = get_color_index(color_sensor.color())
        debug_print("package_color1 is "+ str(package_color1))

        if package_color1 is None:
            ev3.speaker.say("package not detected, retry now")
        else:
            #debug_print("package detected")
            wait(500)
        # second time reading color
        package_color2 = get_color_index(color_sensor.color())
        debug_print("package_color2 is "+ str(package_color2))
        if package_color2 is None:
            if package_color1 is None:
                ev3.speaker.say("package not detected, sleep now")
                wait(100)
                while True:
                    distance = ultrasonic_sensor.distance()
                    if distance < 50:
                        ev3.speaker.say("wake up")
                        wait(100)
                        break
                    wait(500)
            else:
               ev3.speaker.say("package color detected error, retry now")
               wait(1000)
        if say == 1:# tell people whaich package is it
            if package_color1 is package_color2:
                if package_color1 is BLACK :
                    ev3.speaker.say("package confirmed, now deliver to black
house")
                elif package_color1 is BLUE:
                    ev3.speaker.say("package confirmed, now deliver to blue
house")
                elif package_color1 is GREEN:
                    ev3.speaker.say("package confirmed, now deliver to green
house")
                elif package_color1 is YELLOW:
                    ev3.speaker.say("package confirmed, now deliver to yellow
house")
                elif package_color1 is RED:
                    ev3.speaker.say("package confirmed, now deliver to red
house")
                elif package_color1 is WHITE:
                    ev3.speaker.say("package confirmed, now deliver to white
house")
```

```
                elif package_color1 is BROWN:
                    ev3.speaker.say("package confirmed, now deliver to brown
house")
                return package_color1
        else:
            return


def get_color_index(package_color):
    if package_color is Color.BLACK :
        return BLACK
    elif package_color is Color.BLUE:
        return BLUE
    elif package_color is Color.GREEN:
        return GREEN
    elif package_color is Color.YELLOW:
        return YELLOW
    elif package_color is Color.RED:
        return RED
    elif package_color is Color.WHITE:
        return WHITE
    elif package_color is Color.BROWN:
        return BROWN
    else:
        return None

#send data to our python-base sever
def send_status_to_server(data): #data should be string
    print("data is " + data)
    host = "10.93.48.134"
    port = 443                  # The same port as used by the server
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host, port))
    s.send(bytes(data, 'utf-8'))
    count = 0
    s.close()

#get command from our server
def get_commmand():
    host = "10.93.48.134"
    port = 443                  # The same port as used by the server
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host, port))
    s.send(bytes("9", 'utf-8'))
    count = 0
    while True:
        data = s.recv(128)
        if data is not None:
            ackText = data.decode('utf-8')
            print(ackText)
            break
```

```
    s.close()
    return data.decode()


def main():
    '''The main function of our program'''
    # set the console just how we want it
    reset_console()
    set_cursor(OFF)
    set_font('Lat15-Terminus24x12')

    # Initialize the motors.
    right_motor = Motor(Port.D)
    left_motor = Motor(Port.A)
    arm_motor = Motor(Port.B)
    ladar_motor = Motor(Port.C)    #220 to -210

    # Initialize the sensors.
    light_sensor_line = LightSensor(Port.S1)
    light_sensot_landmark = LightSensor(Port.S2)
    ultrasonic_sensor = UltrasonicSensor(Port.S3)
    color_sensor = ColorSensor(Port.S4)

    start(right_motor,left_motor,
arm_motor,ladar_motor,light_sensor_line,light_sensot_landmark, color_sensor,
ultrasonic_sensor)


if __name__ == '__main__':
    main()
```

Appendix G – Web app code

Include the complete code with comment statements for the front end and back end of your web page. If this is going to be too long, only include the code sections that you have written and actively contributed to. Clearly identify each portion of the code for each functionality and file. Use sub-sections, if necessary, to clearly organized different types of code.

Do NOT include links. The objective is to specifically communicate the code that your group actively worked on for the project with comments and documentation on what it does and where it fits.

This is in addition to submitting the complete code file(s). If you do not have a web app, still do include this appendix heading, but indicate "No web app/page was developed".

The code must be readable, with indentation, syntax highlighting (that is, copy with colour-coding), organized, and on white background. The code must be in text (that is, absolutely no snapshots of the code).

Home Page:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta content="width=device-width, initial-scale=1.0" name="viewport">
    <title>Gotham's Delivery Service</title>
    <link rel = "stylesheet" href = style.css>
    <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>

<body>
    <nav id = "navbar">
        <div id = "logo">
            <img src = "shoppingcart.png" alt = "" onclick = "togglemenu()">
        </div>
        <ul>
            <li > <a  href = "home.html">Home</a> </li>
            <li > <a  href = "background.html">AboutUs</a> </li>
            <li > <a  href = "order.html">Order</a> </li>
            <li > <a  href = "login.html">Login</a> </li>
        </ul>
    </nav>
    <section id = "home">
        <h1 class = "h-primary center">Gotham's Delivery Service</h1>
        <p class = "center">The most promising delivery service</p>
        <p class = "center">Fast && Accurate</p>
        <a href = "order.html"><button class = "btn">Order now<img src =
arrow.png></button></a>
```

```
    </section>

    <section id= "services-container">
        <h1 class = "h-primary center">Functionality</h1>
        <div id = "services">
            <div class = "box">
                <img src="shoppingcart.png", alt="">
                <a href = "order.html"><h2 class = "h-secondary
center">Order</h2></a>
                <p class = "center">You can simply choose your destination
and package</p>
            </div>
            <div class = "box">
                <img
src="https://cdn.iconscout.com/icon/free/png-256/order-status-1851726-156938
0.png", alt="">
                <a href = "status.html"><h2 class = "h-secondary
center">Status</h2></a>
                <p class = "center">You can track where the package is and
the status of the car</p>
            </div>
            <div class = "box">
                <img
src="https://upload.wikimedia.org/wikipedia/commons/thumb/a/aa/Google_Maps_i
con_%282020%29.svg/536px-Google_Maps_icon_%282020%29.svg.png", alt="">
                <a href = "map.html"><h2 class = "h-secondary
center">Map</h2></a>
                <p class = "center">You can see the current position of the
car on the map</p>
            </div>
        </div>
    </section>
    <script>
        var menuList = document.getElementById("menuList");
        function togglemenu() {
            if (menuList.style.height == "0px") {
                menuList.style.height = "130px"
            } else {
                menuList.style.height = "0px"
            }
        }

    </script>
</body>

</html>
```

Home Page CSS:

```css
*{
    margin:0px;
    padding:0px;
}

/* navigation bar */
#navbar {
    display:flex;
    align-items: center;
    position: relative;

}
#navbar::before {
    content: "";
    background: black;
    position:absolute;
    width: 100%;
    height: 100%;
    z-index: -1;
    opacity: 0.4;
    top:0px;
    left:0px;
}

/*navigation bar img and logo*/

#logo {
    margin:2px 4px;
}

#logo img{
    margin:2px 10px;
    width:50px;
}

/* navigation bar list styling */
ul {
    display: flex;
    list-style: none;
}

ul li {
    list-style:none;
    font-size: 1.5rem;
```

```
}

ul li a {
    text-decoration:none;
    padding: 8px 18px;
    border-radius: 10px;
    color:black;
}

ul li a:hover{
    color:black;
    background:white;
}

/* home section */
#home{
    display:flex;
    flex-direction: column;
    padding:3px 200px;
    justify-content: center;
    align-items:center;
    height:600px;
    top:0px;
    left:0px;
}
#home::before{
    resize:both;
    content: "";
    position:absolute;
    width:100%;
    height:90%;
    top:0px;
    left:0px;
    background: url("bg.jpg")no-repeat center center/cover;
    z-index: -1;
    opacity:0.6;

}
#home h1{
    color:white;
}
#home p{
    color:white;
    font-size: 2rem;
}

/* service */
#services {
```

```
    display: flex;
    margin: 10px;
}
#services .box {
    margin:7px 7px;
    border:2px solid brown;
    border-radius:25px;
    padding:10px;
}
#services .box img{
    height:100px;
    display:block;
    margin:auto;
    padding:10px;
}

ul li a:hover{
    color:black;
    background:white;
}

/* utility class */
.h-primary {
    padding: 13px;
    font-size: 3.5rem;
}
.h-secondary {
    text-decoration: none;
    padding: 13px;
    font-size: 2rem;
}

.center {
    text-align: center;
}
.btn {
    font-size: 1.5rem;
    border-radius: 10px;
    padding: 5px 25px;
    border: 2px solid brown;
    background: rgb(233,222,222);
    cursor: pointer;
}
```

Order page HTML:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="order.css">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
    <script src="order.js"></script>
    <title>Gotham's Delivery-Ordering Page</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">
</head>

<body>
    <nav id="navbar">
        <div id="logo">
            <img src="shoppingcart.png" alt="">
        </div>
        <ul>
            <li> <a href="home.html">Home</a> </li>
            <li> <a href="background.html">AboutUs</a> </li>
            <li> <a href="order.html">Order</a> </li>
            <li> <a href="login.html">Login</a> </li>
        </ul>
    </nav>
    <section id="home">
        <h1 class="h-primary center">Gotham's Delivery Service</h1>
        <p class="center">This page allows you to order</p>
    </section>

    <div class="Place-order">
        <div class="Delivery__address">

            <input style=" width: 300px;
            padding: 12px 20px;
            margin: 8px 0;
            box-sizing: border-box;" id="Adr" type="" class="" autofocus
placeholder="Enter deliver address">
            <select style="height: 100 px; width: 200px;" id=""
name="Deliver or Pick Up">
                <option value="Deliver">DELIVER</option>
                <option value="Pick up">PICK UP</option>
            </select>
        </div>
    </div>
```

```
    <div class="images">
        <h1>Dear customer, please click your house to make a order</h1>
        <img style="width: 250px; height: 150px; border-radius: 10%;"

src="https://images.pexels.com/photos/106399/pexels-photo-106399.jpeg?auto=c
ompress&cs=tinysrgb&dpr=1&w=500"
            alt="" id="house1">
        <img style="width: 250px; height: 150px ; border-radius: 10%;"

src="https://images.unsplash.com/photo-1568605114967-8130f3a36994?ixlib=rb-1
.2.1&ixid=MnwxMjA3fDB8MHxzZWFyY2h8NXx8aG91c2V8ZW58MHx8MHx8&w=1000&q=80"
            alt="" id="house2">
        <img style="width: 250px; height: 150px; border-radius: 10%;"

src="https://torontolife.com/wp-content/uploads/2019/10/HZIsuuNp.jpg" alt=""
id="house3">

        <img style="width: 250px; height: 150px ; border-radius: 10%;"

src="http://www.contemporist.com/wp-content/uploads/2019/10/modern-house-cov
ered-outdoor-entertaining-021019-840-01-800x533.jpg"
            alt="house 4 pic" id="house4">
    </div>
    <main>
        <section>
            <div class="rule">
                <div class="des1">
                    <h1>
                        House 1
                    </h1>
                </div>
                <div class="des2">
                    <h1>
                        House 2
                    </h1>
                </div>
                <div class="des3">
                    <h1>
                        House 3
                    </h1>
                </div>
                <div class="des4">
                    <h1>
                        House 4
                    </h1>
                </div>
                <h1 class = "des5">Different colour corresponding to
```

```
different colour on the map</h1>
            </div>

        </section>
        <section>
            <div>
                <h3 style="text-align: center;" id="House">Nobody is
shopping???</h3>
            </div>
            <div class="flipper">
                <div class="flip-box">
                    <div class="flip-box-inner">
                        <div class="flip-box-front">
                            <h2>Guess your likes</h2>
                            <img style="width:200px; height: 200px"

src="https://th.bing.com/th/id/OIP.becW_OHFoaHHDrgqK7sdcQHaHa?w=177&h=180&c=
7&r=0&o=5&dpr=2.5&pid=1.7"
                                alt="">
                        </div>
                        <div class="flip-box-back">
                            <p>Steak</p>
                            <img style="border-radius: 100%; width:200px;
height:200px; background-size: cover;
                            border: 6px solid green ;"

src="https://imagesvc.meredithcorp.io/v3/mm/image?url=https%3A%2F%2Fstatic.o
necms.io%2Fwp-content%2Fuploads%2Fsites%2F9%2F2014%2F01%2F10%2F201402-xl-but
ter-basted-rib-eye-steak-2000.jpg&q=60"
                                alt="">
                        </div>
                    </div>
                </div>
                <div class="flip-box1">
                    <div class="flip-box-inner1">
                        <div class="flip-box-front">
                            <h2>Guess your likes</h2>
                            <img style="width:200px; height: 200px"

src="https://th.bing.com/th/id/OIP.becW_OHFoaHHDrgqK7sdcQHaHa?w=177&h=180&c=
7&r=0&o=5&dpr=2.5&pid=1.7"
                                alt="">
                        </div>
                        <div class="flip-box-back">
                            <p>Sashimi</p>
                            <img style="border-radius: 100%; width:200px;
height:200px; background-size: contain;
                            border: 6px solid green ;"
```

```
src="https://digitalmarketing.blob.core.windows.net/13214/images/items/image
730197.png"
                                alt="">
                        </div>
                    </div>
                </div>
                            </div>
            <div>
                <input id="searchinput" class="searchinput" autofocus
placeholder="What do u like to search?" />
                <a href="http://google.com"><button
id="searchBtn">SEARCHHHH</button></a>

                                <input id="input" autofocus placeholder="What
do u like to order?" />
                <!-- <button id="submit1">Log in</button> -->
                <button id="submit">Submit</button>
                <button id="remove">Remove</button>
                <div id="content">
                    <div id="notes"></div>
                </div>
                <p id="errormess"></p>
                <br>
                <div>
                    <label for="food">Choose deliver type:</label>
                    <select name="food" id="food">
                        <option value="cooked">Cooked Food</option>
                        <option value="grocery">Grocery</option>
                        <option value="essentials">Daily Essentials</option>
                        <option value="drinks">Drinks</option>
                    </select>
                    <button id="ss">buy</button>
                    <p id="testing1"></p>
                </div>
                <script>

                </script>
            </div>

            <!-- history below -->
            <h2>It's the ordering history below</h2>
            <div class="History" id="History">

            </div>

        </section>
```

```
    </main>

</body>

</html>
```

Order CSS:

```
*{
    margin:0px;
    padding:0px;
}

/* navigation bar */
#navbar {
    display:flex;
    align-items: center;
    position: relative;

}
#navbar::before {
    content: "";
    background: black;
    position:absolute;
    width: 100%;
    height: 100%;
    z-index: -1;
    opacity: 0.4;
    top:0px;
    left:0px;
}


/*navigation bar img and logo*/

#logo {
    margin:2px 4px;
}

#logo img{
    margin:2px 10px;
    width:50px;
```

```
}

/* navigation bar list styling */
ul {
    display: flex;
    list-style: none;
}

ul li {
    list-style:none;
    font-size: 1.5rem;
}

ul li a {
    text-decoration:none;
    padding: 8px 18px;
    border-radius: 10px;
    color:black;
}

ul li a:hover{
    color:black;
    background:white;
  }


/* home section */
#home{
    display:flex;
    flex-direction: column;
    padding:3px 200px;
    justify-content: center;
    align-items:center;
    height:600px;
    top:0px;
    left:0px;
}
#home::before{
    resize:both;
    content: "";
    position:absolute;
    width:100%;
    height:101%;
    top:0px;
    left:0px;
    background: url("bg.jpg") no-repeat center center/cover;
    z-index: -1;
    opacity:0.6;
```

```
}
#home h1{
    color:white;
}
#home p{
    color:white;
    font-size: 2rem;
}
.h-primary {
    padding: 13px;
    font-size: 3.5rem;
}
.center {
    text-align: center;
}

.Delivery__address {
    display: contents;
    width: 10%;
    height: 10%;
    top: 10%;
    margin: 0 auto;
    position: relative;
}


/* utility class */
.h-primary {
    padding: 13px;
    font-size: 3.5rem;
}

.btn {
    font-size:1.5rem;
    width: 240px;
    border:0;
    padding: 12px 10px;
    outline:none;
    color:white;
    background: linear-gradient(to right, #fb5283, #ff3527);
    border-radius: 6px;
    cursor:pointer;
    transition: width 0.5s;
}
.btn img {
    width: 30px;
    display:none;
```

```css
}

.btn:hover img {
    display: block;
}

.btn:hover {
    width:250px;
    display: flex;
    align-items:center;
    justify-content: space-between;
}

#Adr,
select {
  width: 10%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}

.rule {
    background-color: #1780ad;
    display: flex;
  }

  .rule .des1,
  .rule .des2,
  .rule .des3,
  .rule .des4 {
    display: flex;

  }
  .rule .des1 h1 {
    background-color: rgb(212, 59, 59);
    margin: 10px;
    padding: 20px;
    font-size: 30px;
  }
  .rule .des2 h1 {
    background-color: rgb(59, 61, 212);
    margin: 10px;
    padding: 20px;
    font-size: 30px;
  }
```

```css
.rule .des3 h1 {
  background-color: rgb(84, 212, 59);
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}
.rule .des4 h1 {
  background-color: burlywood;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}

/* filpbox */
.flipper {
  display: flex;
  background-color: greenyellow;
}
.flip-box,
.flip-box1,
.flip-box2 {
  display: flex;
  padding: 10px;
  background-color: transparent;
  width: 400px;
  height: 300px;
  border: 1px solid #f1f1f1;
  perspective: 1000px;
}

.flip-box-inner,
.flip-box-inner1,
.flip-box-inner2 {
  position: relative;
  width: 100%;
  height: 100%;
  text-align: center;
  transition: transform 0.8s;
  transform-style: preserve-3d;
}

.flip-box:hover .flip-box-inner,
.flip-box1:hover .flip-box-inner1,
.flip-box2:hover .flip-box-inner2 {
  transform: rotateX(180deg);
}

.flip-box-front,
```

```css
.flip-box-back {
  position: absolute;
  width: 100%;
  height: 100%;
  -webkit-backface-visibility: hidden;
  backface-visibility: hidden;
}

.flip-box-front {
  /* background-color: #bbb; */
  padding: 10px;
  color: black;
}

.flip-box-back {
  /* background-color: dodgerblue; */
  padding: 10px;
  color: white;
  transform: rotateX(180deg);
}

#food {
  width: 200px;
}


#cf1,
#cf2,
#cf3 {
  display: block;
  width: 200px;
  height: 200px;
  background-size: contain;
}
#dr1,
#dr2,
#dr3 {
  display: block;
  width: 200px;
  height: 200px;
  background-size: contain;
}
#gr1,
#gr2,
#gr3 {
  display: block;
  width: 200px;
  height: 200px;
```

```
  background-size: contain;
}

#ess1,
#ess2,
#ess3 {
  display: block;
  width: 200px;
  height: 200px;
  background-size: contain;
}
.imagesssss {
  display: flex;
}

@media screen and (max-width:700px) {

  nav ul {
      display: block;
      width:100%;
      position: absolute;
      top: 50px;
      margin: 0px;
      left: 0;
      z-index: 2;
  }


  nav ul li {
      /* color: #fff; */
      display: block;
      margin-top: 10px;
      margin-bottom: 10px;
  }

  #logo img{
      display:block;
  }

  #home p{
      display:none;
  }

  .rule{
      display: block;
      padding:0px;
      margin:0px;
  }
```

```
    .rule .des5{
        display:none;
    }

}
```

Status HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="status.css">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
    <script src="status.js"></script>
    <title >Gotham's Delivery-Ordering Page</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">

</head>

<body>
    <nav id="navbar">
        <div id="logo">
            <img src="shoppingcart.png" , alt="">
        </div>
        <ul>
            <li> <a href="home.html">Home</a> </li>
            <li> <a href="background.html">About Us</a> </li>
            <li> <a href="OrderSimon.html">Order</a> </li>
            <li> <a href="login.html">Login</a> </li>
        </ul>
    </nav>
    <section id="home">
        <h1 style="color: black" class="h-primary center">Gotham's Delivery
Service</h1>
        <p  style="color: black" class="center">Display the status of your
order</p>
        <a href="order.html"><button style="background-color: #4CAF50; /*
Green */
            border: none;
            color: white;
```

```
            padding: 20px;
            text-align: center;
            text-decoration: none;
            display: inline-block;
            font-size: 16px;
            margin: 4px 2px;
            cursor: pointer;
            border-radius: 12px;" class="btn">Order now</button></a>
    </section>
    <div class="tableofStatus">
        <table>
            <tr>
                <th>Location</th>
                <th>Package Color</th>
                <th>Estimated Delivery Time</th>
            </tr>
            <tr>
                <td id="td11"></td>
                <td id="td12"></td>
                <td id="td13"></td>
            </tr>
            <tr>
                <td id="td21"></td>
                <td id="td22"></td>
                <td id="td23"></td>
            </tr>
            <tr>
                <td id="td31"></td>
                <td id="td32"></td>
                <td id="td33"></td>
            </tr>
        </table>
    </div>

    <div>
        <div class="carinfo">
            <div class="carinfo-left">
                <img style="height : 360px; width:620px" id="audi"

src="https://d2nmr6p48f8xwg.cloudfront.net/content_pictures/pictures/000/001
/526/79583159a0a854f153e7f969e5b4916c57a51ae2Lego-Mindstorms-Ev3-Car-With-Tw
o-Driving-Large-Motors-Fllcasts.png"
                    alt="">
            </div>
            <div class="carinfo-right">
                <h3>We are proud to deliver your package via the eletrical
and sustainable EV3 LEGO CAR</h6>
            </div>
```

```
        </div>

        <div class="container">
            <h2>Basic Progress Bar</h2>
            <div class="progress" id = "progess">
                <div id ="bar1" class="progress-bar" role="progressbar"
aria-valuenow="70" aria-valuemin="0" aria-valuemax="100"
                    style="width:0%">
                    <span class="sr-only" id="precent">0% Complete</span>
                </div>
            </div>
        </div>
        <a href="status.html"><button type="button" class="btn btn-success
text-center"  id="freshhhhhhh" onclick="xx()" >update progress</button></a>

</body>

</html>
```

Status CSS:

```
*{
    margin:0px;
    padding:0px;
}

/* navigation bar */
#navbar {
    display:flex;
    align-items: center;
    position: relative;

}
#navbar::before {
    content: "";
    background: black;
    position:absolute;
    width: 100%;
    height: 100%;
    z-index: -1;
    opacity: 0.4;
    top:0px;
    left:0px;
}

/*navigation bar img and logo*/
```

```css
#logo {
    margin:2px 4px;
}

#logo img{
    margin:2px 10px;
    width:50px;
}

/* navigation bar list styling */
ul {
    display: flex;
    list-style: none;
}

ul li {
    list-style:none;
    font-size: 1.5rem;
}

ul li a {
    text-decoration:none;
    padding: 8px 18px;
    border-radius: 10px;
    color:black;
}

ul li a:hover{
    color:black;
    background:white;
}


/* home section */
#home{
    display:flex;
    flex-direction: column;
    padding:3px 200px;
    justify-content: center;
    align-items:center;
    height:600px;
    top:0px;
    left:0px;
}
#home::before{
    resize:both;
    content: "";
```

```
    position:absolute;
    width:100%;
    height:90%;
    top:0px;
    left:0px;
    background: url("bg.jpg") no-repeat center center/cover;
    z-index: -1;
    opacity:0.6;

}
#home h1{
    color:white;
}
#home p{
    color:white;
    font-size: 2rem;
}

/*
table */
th,
td {
  border: 1px solid black;
  border-style: solid;
  border-color: #96d4d4;
  border-radius: 10px;
  background-color: #96d4d4;
  padding-top: 10px;
  padding-bottom: 20px;
  padding-left: 30px;
  padding-right: 40px;
}
table,
th,
td {
  border: 1px solid white;
  border-collapse: collapse;
}

table {
  width: 100%;
}

/* car image and car description */
.carinfo-right:first-child {
  margin-right: 40px;
  padding-right: 20px;
}
```

```css
.carinfo-right {
  background-color: #20ca29;
  padding-right: 250px;
  justify-content: center;
  padding-left: 20px;
}
.carinfo-left {
  flex: 1;
  background-color: #d4bb96;
}

.carinfo {
  display: flex;
  flex-direction: row;
}

/* utility */
.h-primary {
    padding: 13px;
    font-size: 3.5rem;
}
.h-secondary {
    text-decoration: none;
    padding: 13px;
    font-size: 2rem;
}

.center {
    text-align: center;
}
.btn {
    font-size: 1.5rem;
    border-radius: 10px;
    padding: 5px 25px;
    border: 2px solid brown;
    background: rgb(233,222,222);
    cursor: pointer;
}
```

Map HTML:

```html
<!DOCTYPE html><html><head></head>

<body>


  <meta charset="UTF-8">
```

```
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="map.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></sc
ript>
  <!-- <script src="map.js"></script> -->
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <title>Gotham's Delivery-Ordering Page</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.cs
s" rel="stylesheet">


  <nav id="navbar">
    <div id="logo">
      <img src="shoppingcart.png" alt="" onclick="togglemenu()">
    </div>
    <ul>
      <li> <a href="home.html">Home</a> </li>
      <li> <a href="background.html">AboutUs</a> </li>
      <li> <a href="order.html">Order</a> </li>
      <li> <a href="login.html">Login</a> </li>
    </ul>
  </nav>
  <section id="home">
    <h1 class="h-primary center">Gotham's Delivery Service</h1>
    <p class="center">The most promising delivery service</p>
    <p class="center">Fast &amp;amp;&amp;amp; Accurate</p>
    <a href="order.html"><button class="btn">Order now<img
src="arrow.png"></button></a>
  </section>

  <br><br><br><br>
  <img id="mapgif" src="map5.gif" usemap="#planetmap">

  <map name="planetmap">
    <area shape="rect" coords="0,180,120,260" alt="house1" href="#house1">
    <area shape="rect" coords="330,600,390,680" alt="house2" href="#house2">
    <area shape="circle" coords="124,58,8" alt="house3" href="#house3">
    <area shape="rect" coords="0,0,82,126" alt="house4" href="#house4">
    <area shape="rect" coords="0,0,82,126" alt="house5" href="#warehouse">
  </map>
</body></html>
```

Map CSS:

```css
*{
  margin:0px;
  padding:0px;
}

#navbar {
  display:flex;
  align-items: center;

}
#navbar::before {
  content: "";
  background: black;
  position:absolute;
  width: 100%;
  height: 100%;
  z-index: -1;
  opacity: 0.4;
  top:0px;
  left:0px;
}

/*navigation bar img and logo*/

#logo {
  margin:2px 4px;
}

#logo img{
  margin:2px 10px;
  width:50px;
}

/* navigation bar list styling */
ul {
  display: flex;
  list-style: none;
}

ul li {
  list-style:none;
  font-size: 1.5rem;
}

ul li a {
  text-decoration:none;
  padding: 8px 18px;
  border-radius: 10px;
```

```
  color:white;
}

ul li a:hover{
  color:black;
  background:white;
}


ul li a:hover{
  color:black;
  background:white;
}

/* home section */
#home{
  display:flex;
  flex-direction: column;
  padding:3px 200px;
  justify-content: center;
  align-items:center;
  height:600px;
  top:0px;
  left:0px;
}
#home::before{

  background: url("bg.jpg")no-repeat center center/cover;
  resize:both;
  content: "";
  position:absolute;
  width:100%;
  height:100%;
  top:0px;
  left:0px;
  z-index: -1;
  opacity:0.7;

}
#home h1{
  position: relative;
  color:white;
}
#home p{
  font-weight: bold;
  color:white;
  font-size: 2rem;
  padding-bottom: 10px;
```

```
}

  #mapgif{
    display: flex;
    align-items: center;
    position: relative;
  }

  .h-primary {
    padding: 13px;
    font-size: 3.5rem;
}
.h-secondary {
    text-decoration: none;
    padding: 13px;
    font-size: 2rem;
}

.center {
    text-align: center;
    position: relative;
}
.btn {
    font-size:1.5rem;
    width: 240px;
    border:0;
    padding: 12px 10px;
    outline:none;
    color:white;
    background: linear-gradient(to right, #fb5283, #ff3527);
    border-radius: 6px;
    cursor:pointer;
    transition: width 0.5s;
    /* font-size: 1.5rem;
    border-radius: 10px;
    padding: 5px 25px;
    border: 2px solid brown;
    background: rgb(233,222,222);
    cursor: pointer;
    transition: width 0.5; */
}

.btn img {
    width: 30px;
    display:none;
}

.btn:hover img {
```

```
    display: block;
}

.btn:hover {
    width:250px;
    display: flex;
    align-items:center;
    justify-content: space-between;
}

@media screen and (max-width:700px) {

    nav ul {
        display: block;
        width:100%;
        position: absolute;
        top: 50px;
        margin: 0px;
        left: 0;
        z-index: 2;
    }


    nav ul li {
        /* color: #fff; */
        display: block;
        margin-top: 10px;
        margin-bottom: 10px;
    }

    #logo img{
        display:block;
    }

}
```

Server JS(Backend):

```
const http = require('http');
const fs = require('fs');
const path = require('path');
const port = 80;

//creating a node js server and allow all html/css/js/images to display on
```

```
the vm
const server = http.createServer((req, res) => {
    let filePath = path.join(
        __dirname,
        "",
        req.url === "/" ? "home.html" : req.url
    );

    let extName = path.extname(filePath);
    let contentType = 'text/html';

    switch (extName) {
        case '.css':
            contentType = 'text/css';
            break;
        case '.js':
            contentType = 'text/javascript';
            break;
        case '.json':
            contentType = 'application/json';
            break;
        case '.png':
            contentType = 'image/png';
            break;
        case '.jpg':
            contentType = 'image/jpg';
            break;
    }

    console.log(`File path: ${filePath}`);
    console.log(`Content-Type: ${contentType}`)

    res.writeHead(200, {'Content-Type': contentType});

    const readStream = fs.createReadStream(filePath);
    readStream.pipe(res);
});

server.listen(port, (err) => {
    if (err) {
        console.log(`Error: ${err}`)
    } else {
        console.log(`Server listening at port ${port}...`);
    }
});
```

Order.js:

```
window.onload = function () {

    var inn = document.getElementById("searchinput");
    var cf1 = document.getElementById("cf1")
    var cf2 = document.getElementById("cf2")
    var cf3 = document.getElementById("cf3")
    var gr1 = document.getElementById("gr1")
    var gr2 = document.getElementById("gr2")
    var gr3 = document.getElementById("gr3")
    var ess1 = document.getElementById("ess1")
    var ess2 = document.getElementById("ess2")
    var ess3 = document.getElementById("ess3")
    var dr1 = document.getElementById("dr1")
    var dr2 = document.getElementById("dr2")
    var dr3 = document.getElementById("dr3")


    // house1234 are four images listening to an onclick event
    var tempHouse = "bad";
    var house1 =document.getElementById("house1");
    house1.addEventListener("click", function(){
        document.getElementById("House").innerHTML= "House 1 is shopping
now";
        tempHouse = "House 1 ordered: ";
    })
    var house2 =document.getElementById("house2");
    house2.addEventListener("click", function(){
        document.getElementById("House").innerHTML= "House 2 is shopping
now";
        tempHouse = "House 2 ordered: ";
    })
    var house3 =document.getElementById("house3");
    house3.addEventListener("click", function(){
        document.getElementById("House").innerHTML= "House 3 is shopping
now";
        tempHouse = "House 3 ordered: ";
    })
    var house4 =document.getElementById("house4");
    house4.addEventListener("click", function(){
        document.getElementById("House").innerHTML= "House 4 is shopping
now";
        tempHouse = "House 4 ordered: ";
    })

    var submitBtn = document.getElementById("submit");
    var inputElement = document.getElementById("input");
```

```
    var notesElement = document.getElementById("notes");
    var removElement = document.getElementById("remove");

    // if a submit btn is clicked, a reminder to display for the user to
pick a house
    // after picking a house by clicking the imgs, input will add to the div
list
    submitBtn.addEventListener("click", function () {

        var div = document.createElement("div");
        div.setAttribute("class", "note");
        if(tempHouse === "bad"){
            document.getElementById("errormess").innerHTML="Please choose a
house first";
            inputElement.value = "";
        }
        else{
            document.getElementById("errormess").innerHTML="";
            div.innerText = tempHouse + inputElement.value;
            notesElement.appendChild(div);
            inputElement.value = "";
            tempHouse="bad"
        }

    })

    // if removeElement is clicked, the child of the div list will be
removed
    removElement.addEventListener("click", function () {
        //notesElement.removeChild();
        if (notesElement.hasChildNodes()) {
            notesElement.removeChild(notesElement.children[0]);
        }
    })

    if(document.getElementById("food").innerHTML == "Daily Essentials"){
        document.getElementById("testing1").innerHTML = " is changing";
    }
    else{
        document.getElementById("testing1").innerHTML = "";
    }
    // select list
    const btn2 = document.querySelector('#ss');
    const sb2 = document.querySelector('#food');
    var historyElement = document.getElementById("History");

    // btn to add text to list
    btn2.onclick = (event) => {
```

```
        event.preventDefault();
        var div = document.createElement("div");
        div.setAttribute("class", "note");
        var divHistory = document.createElement("div");
        divHistory.setAttribute("class", "History");
        if(tempHouse === "bad"){
            document.getElementById("testing1").innerHTML="Please choose a
house first";
        }
        else{
            document.getElementById("testing1").innerHTML="";
            if (sb2.selectedIndex == 0) {
                document.getElementById("testing1").innerHTML =  tempHouse +
"cooked food";
                cf1.style.width= "200px";
                cf2.style.width= "200px";
                cf3.style.width= "200px";

                dr1.style.width= "0px";
                dr2.style.width= "0px";
                dr3.style.width= "0px";
                gr1.style.width= "0px";
                gr2.style.width= "0px";
                gr3.style.width= "0px";
                ess1.style.width= "0px";
                ess2.style.width= "0px";
                ess3.style.width= "0px";
            }
            else if (sb2.selectedIndex == 1) {
                document.getElementById("testing1").innerHTML = tempHouse +
"grocery";
                cf1.style.width= "0px";
                cf2.style.width= "0px";
                cf3.style.width= "0px";
                dr1.style.width= "0px";
                dr2.style.width= "0px";
                dr3.style.width= "0px";
                gr1.style.width= "200px";
                gr2.style.width= "200px";
                gr3.style.width= "200px";
                ess1.style.width= "0px";
                ess2.style.width= "0px";
                ess3.style.width= "0px";
            }
            else if (sb2.selectedIndex == 2) {
                document.getElementById("testing1").innerHTML = tempHouse +
"daily essentials";
                cf1.style.width= "0px";
```

```
                cf2.style.width= "0px";
                cf3.style.width= "0px";
                dr1.style.width= "0px";
                dr2.style.width= "0px";
                dr3.style.width= "0px";
                gr1.style.width= "0px";
                gr2.style.width= "0px";
                gr3.style.width= "0px";
                ess1.style.width= "200px";
                ess2.style.width= "200px";
                ess3.style.width= "200px";
            }
            else if (sb2.selectedIndex == 3) {
                document.getElementById("testing1").innerHTML = tempHouse +
"drinks";
                cf1.style.width= "0px";
                cf2.style.width= "0px";
                cf3.style.width= "0px";
                dr1.style.width= "200px";
                dr2.style.width= "200px";
                dr3.style.width= "200px";
                gr1.style.width= "0px";
                gr2.style.width= "0px";
                gr3.style.width= "0px";
                ess1.style.width= "0px";
                ess2.style.width= "0px";
                ess3.style.width= "0px";
            }
            // notes
            div.innerText = document.getElementById("testing1").innerHTML;
            notesElement.appendChild(div);
            // history
            divHistory.innerText =
document.getElementById("testing1").innerHTML;
            historyElement.appendChild(divHistory);
            tempHouse="bad";
        }

    };



}
```

Status.JS:

```
window.onload = function () {


    // use ajax to read Json file
    // if package is not delivered then check location then diplay info
    // if package is delivered display corresponding info
    function xx() {
        console.log("11")
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function () {
            if (this.readyState == 4 && this.status == 200) {
                // document.getElementById("aa").innerHTML =
                //     this.responseText;
                var jsonParsedLocation = JSON.parse(this.responseText);
                if (jsonParsedLocation.package_delivered == 0) {

                    console.log("12")
                    if (jsonParsedLocation.location == 1) {
                        document.getElementById("bar1").style.width = "20%";
                        document.getElementById("bar1").innerHTML = "20%
complete";
                        document.getElementById("td11").innerHTML = "Your
delivery driver has your order";
                        document.getElementById("td13").innerHTML = "1:00";
                        document.getElementById("td12").innerHTML =
jsonParsedLocation.package_color;
                        console.log(jsonParsedLocation.package_color);
                        console.log("13")
                    }
                    else if (jsonParsedLocation.location == 2) {
                        document.getElementById("bar1").style.width = "40%";
                        document.getElementById("bar1").innerHTML = "40%
complete";
                        document.getElementById("td11").innerHTML = "Your
delivery driver is on his/her way";
                        document.getElementById("td13").innerHTML = "00:45";
                        document.getElementById("td12").innerHTML =
jsonParsedLocation.package_color;

                    }
                    else if (jsonParsedLocation.location == 3) {
                        document.getElementById("bar1").style.width = "60%";
                        document.getElementById("bar1").innerHTML = "60%
complete";
                        document.getElementById("td11").innerHTML = "Your
delivery driver is almost arrived";
                        document.getElementById("td13").innerHTML = "00:30";
                        document.getElementById("td12").innerHTML =
```

```
jsonParsedLocation.package_color;

                    }
                    else if (jsonParsedLocation.location == 4) {
                        document.getElementById("bar1").style.width = "80%";
                        document.getElementById("bar1").innerHTML = "80%
complete";
                        document.getElementById("td11").innerHTML = "Please
prepare for the arrival";
                        document.getElementById("td13").innerHTML = "00:15";
                        document.getElementById("td12").innerHTML =
jsonParsedLocation.package_color;
                    }
                }
                else if (jsonParsedLocation.package_delivered == 1) {
                    document.getElementById("bar1").style.width = "100%";
                    document.getElementById("bar1").innerHTML = "100%
complete";
                    document.getElementById("td11").innerHTML = "Enjoy your
package";
                    document.getElementById("td13").innerHTML = "00:00";
                    document.getElementById("td12").innerHTML =
jsonParsedLocation.package_color;
                }
                else {
                    console.log("huh?????");
                }
                console.log(jsonParsedLocation);
                console.log(jsonParsedLocation.location);
            }
        };
        xhttp.open("GET", "status2.json", true);
        xhttp.send();

    }

    xx();
}
```

Map.js(backend):

```
var fs = require("fs");

var cheerio = require("cheerio");
```

```
// while(true){


var htmlPath = './map.html';

var outPath = './map.html';

var replaceValues = require('./status2.json');

fs.readFile(htmlPath, { encoding: 'utf-8' }, function (err, html) {
  const $ = cheerio.load(html);
  if (err) {
  } else {
    for (var key in replaceValues) {
      if (key == "location") {
        if (replaceValues[key] == 1) {
          $("img").each(function () {
            var id = $(this).attr("id");
            var url = "map1.gif";
            var new_href = encodeURIComponent(url);
            if (id == "mapgif") {
              $(this).attr("src", new_href);
            }
          });
          fs.writeFile(outPath, $.html(), function (err) {
            if (err) {
              throw err;
            }
          });
        }

        if (replaceValues[key] == 2) {
          $("img").each(function () {
            var id = $(this).attr("id");
            var url = "map2.gif";
            var new_href = encodeURIComponent(url);
            if (id == "mapgif") {
              $(this).attr("src", new_href);
            }
          });
          fs.writeFile(outPath, $.html(), function (err) {
            if (err) {
              throw err;
            }
          });
        }

        if (replaceValues[key] == 3) {
```

```
  $("img").each(function () {
    var id = $(this).attr("id");
    var url = "map3.gif";
    var new_href = encodeURIComponent(url);
    if (id == "mapgif") {
      $(this).attr("src", new_href);
    }
  });
  fs.writeFile(outPath, $.html(), function (err) {
    if (err) {
      throw err;
    }
  });
}

if (replaceValues[key] == 4) {
  $("img").each(function () {
    var id = $(this).attr("id");
    var url = "map4.gif";
    var new_href = encodeURIComponent(url);
    if (id == "mapgif") {
      $(this).attr("src", new_href);
    }
  });
  fs.writeFile(outPath, $.html(), function (err) {
    if (err) {
      throw err;
    }
  });
}

if (replaceValues[key] == 5) {
  $("img").each(function () {
    var id = $(this).attr("id");
    var url = "map5.gif";
    var new_href = encodeURIComponent(url);
    if (id == "mapgif") {
      $(this).attr("src", new_href);
    }
  });
  fs.writeFile(outPath, $.html(), function (err) {
    if (err) {
      throw err;
    }
  });
}

if (replaceValues[key] == 0) {
```

```
            $("img").each(function () {
              var id = $(this).attr("id");
              var url = "mapwait.gif";
              var new_href = encodeURIComponent(url);
              if (id == "mapgif") {
                $(this).attr("src", new_href);
              }
            });
            fs.writeFile(outPath, $.html(), function (err) {
              if (err) {
                throw err;
              }
            });
          }

        }

      }
    }
});
```

Appendix I – Other code

Code forcho_server.py

```
import os
import socket
from time import sleep
import re

if __name__ == "__main__":
    color_dict = {
        "blue": 0,
        "green": 1,
        "yellow": 2,
        "black": 5,
        "white": 6,
        "red": 7,
        "brown": 8,
        "auto": 9
    }
    me_dict = {
        "pickup": 0,
        "wait": 1,
    }
    # Create a stream based socket(i.e, a TCP socket)
    # operating on IPv4 addressing scheme
    serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # Bind and listen
    serverSocket.bind(("10.93.48.134", 443))
    serverSocket.listen()
    # Accept connections
    serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    # Bind and listen
    serverSocket.bind(("10.93.48.134", 443))
    serverSocket.listen()
    # Accept connections
    old_job_id = -1
    destination = "auto"
    deliver_method = "pickup"
    while (True):
        os.system("sudo node ordertransmit.js")
        os.system("sudo node map.js")
        #listen to the port if any successful connection
        (clientConnected, clientAddress) = serverSocket.accept()
        print("Accepted a connection request from %s:%s" % (clientAddress[0],
clientAddress[1]))
        dataFromClient = clientConnected.recv(1024)
        recieve = dataFromClient.decode()
        print(recieve)
        #determine if client want to command or sending command
        if "9" not in recieve:
```

```
            command = "python3 write.py " + recieve
            print(command)
            os.system(command)
        elif "9" in recieve:
            # Send some data back to the client
            f = open("secret.html", "r")
            new_job_id = 0
            # sending command to the robot base on the website request
            for line in f:
                line = line.strip()
                line = re.sub('<[^<]+?>', '', line)
                print(line)
                if "house:" in line:
                    destination = line.replace("house:", "")
                elif "deliver_method:" in line:
                    deliver_method = line.replace("deliver_method:", "")
                    print("deliver_method is" + deliver_method)
                elif "jobid:" in line:
                    new_job_id = line.replace("jobid:", "")
            if old_job_id != new_job_id: #if job id is same then means this job
is alreay done
                old_job_id = new_job_id
                command = str(color_dict[destination]) +
str(me_dict[deliver_method])
                clientConnected.send(command.encode())
            else:
                command = str('70')
                clientConnected.send(command.encode())
        sleep(0.5)
#ps -fA | grep python
```

Code for write.py

```
import sys
# change the json file for our website to read, mainly used for updating the map
and status

def main(argv):
    color_dict = {
        '0': "BLUE",
        '1': "GREEN",
        '2': "YELLOW",
        '5': "BLACK",
        '6': "WHITE",
        '7': "RED",
        '8': "BROWN",
        '9': "AUTO"
    }
    # first digit mean package_delivered, 0 means not delivered, 1 means
delivered
    # second digit mean robot location,
    f = open('status2.json', 'w')

    #arg[0] = package_delivered
    #arg[1] = location of robot
    #arg[2] = package_color

    result_string = "{\"package_delivered\":" + argv[0] + ", \"location\":" +
argv[1] + ", \"package_color\":\"" + color_dict[argv[2]]+"\"}"
    f.write(result_string)
    f.close()

if __name__ == '__main__':
    main(sys.argv[1])
```