



# **Brain Controlled Wheelchair**

## **2 Executive Summary :**

## 2.1 :

Independent mobility is a necessity to live everyday life for human beings. A person with physical challenges has restricted mobility. For these people, Brain-Computer Interface (BCI) provides a promising solution. Using

Electroencephalogram (EEG) signals for movement of a wheelchair the mobility of these persons can be improved. The proposed system is based on the Artificial Neural Network (ANN) algorithm. We will present Mu rhythm signals that provide commands to a wheelchair. Using wireless links between headgear and computer, commands to control the wheelchair can be issued. The immediate mission is to offer basic communication skills to these patients , who may be totally disabled or “locked in “ , so that they may communicate their needs to caregivers or even function word processing systems or neurotheses. BCI systems can eventually provide an essential new communication and control option for people with motor disabilities , as well as a supplementary control channel or a control channel useful in specific circumstances for people without disabilities.

## 2.2 Detailed explanation of all the innovative ideas and features that the team has

## **included in the project and a critical discussion of the merits of such innovations.**

We introduce a product that will change our stakeholder's life completely, with help our system.

**(1)** Patient will be able to move his chair only by his thoughts.

**(2)** Our app will be connected with the electroencephalography (EEG) sensor that detect patient's brain waves, brain waves can be used to store patient's mood and some health information. **(3)** It can also be used as an input to Arduino board that controls motors that move the chair, detected brain waves can also tell us that patient is blinking.

**(4)** We will use patient's eye blinking to enable him to select any items in our app.

Our mobile Application (APP) has 2 separate user interfaces:

- Patient interface:

- (1)** Can ask for help by selecting from some built-in like (ask for food, ask for WC, ask for water, etc.) only using his eye blinked.

- (2)** Patient can also follow up his health information.

- Escort interface:

- (1)** Escort will fill in some needed patient's information.

- (2)** Can track patient's mood and health information.

- (3)** Manage his profile and order any technical support.





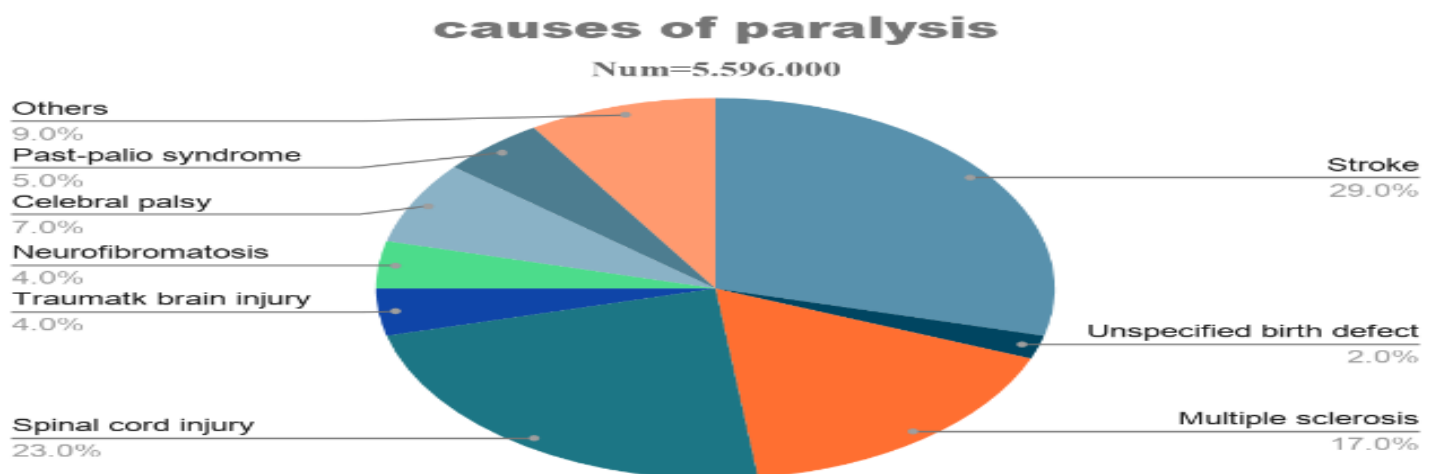
## 4 Introduction and Background Review

### 4.1 Problem Statement

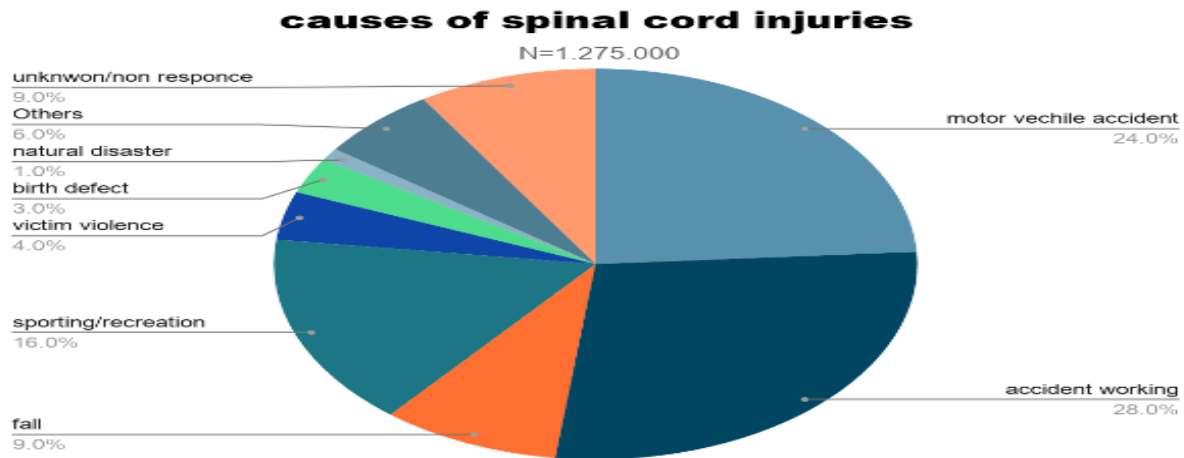
As a result of a variety of accidents or diseases, such as a spinal cord injury (SCI) or amyotrophic lateral sclerosis (ALS), muscular dystrophies, and multiple sclerosis. Individuals suffering from severe movement disorders such as ALS or Locked in Syndrome, is a condition in which a patient is awake and aware of his or her surroundings but is unable to communicate or perform any action due to paralysis of almost all voluntary muscles in the body (with the exception of eye movements and blinking), consider these patients to have a fully functional brain trapped inside a non-functional body, these people are forced to accept a lower standard of living. Relying on the care of other people.

As a result, there were requirements to call for a novel interface to assist the disabled in living a better life and being completely free to move and express themselves .

### 4.2 Injury and disease statistics



(Fig.4.1) Analysis of The Cases of Paralysis



**(Fig.4.2) Analysis of The Cases of Spinal Injuries**

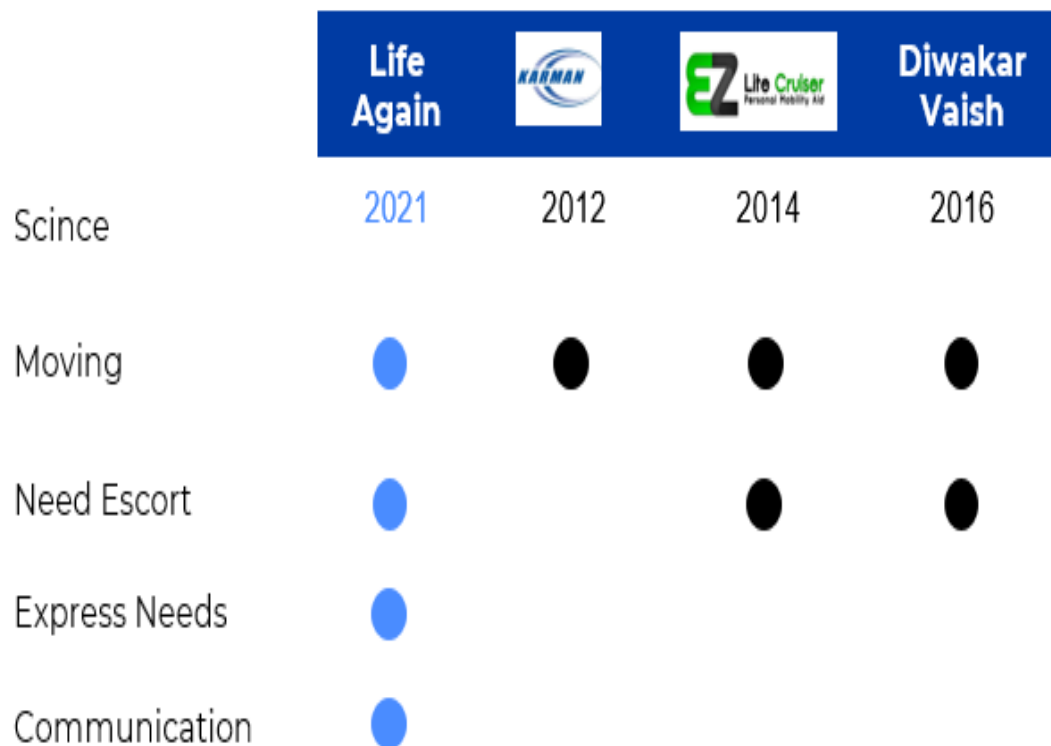
### 4.3 Current Solutions (Background Review)

With the increase in the number of patients with ALS, there must be a technological solution to this problem. We introduce a product that will change our stakeholder's life completely, with help our system.

- (1) Patient will be able to move his wheelchair by thinking.
- (2) Our app will be connected with the electroencephalography (EEG) sensor that detect patient's brain waves, brain waves can be used to store patient's mood and some health information.
- (3) It can also be used as an input to Arduino board that controls motors that move the chair, detected brain waves can also tell us that patient is blinking.
- (4) We will use patient's eye blinking to enable him to select any items in our app.
- (5) We will use patient's eye blinking to enable to typing.
- (6) He can make emergency calls. Product that will completely change the lives of its stockholders, convenient wheelchairs require patients to be escorted by someone to move, our Product allow patients to move on their own, not only that, but he can also communicate with the rest of the world by writing and making emergency calls and express their opinion & feelings like healthy people.

#### 4.4 competitors

We have more than one competing company. We tried to look at our prominence in Life Again from these companies with several features such as (Scince,Moving,Need Escort,Express Need, Communication).



(Fig.4.3) competitors



## **5) System Description :**

### **1.1 System Requirements :**

System requirements are the needed configurations for the system to operate efficiently. The next three subsections will discuss the functional requirements, non-functional requirements, and user requirements.

#### **5.1.1 Functional Requirements**

In systems engineering, functional requirements are directly concerned with the system services, where a function is described as a specification of behavior. In this subsection, we list the functions required in our system. We, also, provide a description for each function. Table 5.1 shows the functional requirements for our mind wave wheel chair system.

#### **5.1.2 Non Functional Requirements**

The system needs to operate efficiently and meet the requirements. Any failure of the components of the systems may lead to one or more of functions to stop or be misused. A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system. Table 5. 2 shows the functional requirements for our mind wave wheel chair system.

(Table 5.1) Functional Requirements for System

#	Functional Requirement	Description
1	<b>Moving and control wheelchair</b>	The patient can move and control the wheelchair through a mind wave sensor. When he wants to move the wheelchair is moving.
2	<b>Eye detection and tracking</b>	The application can detect and track the eye blinking of the patient's eyes through the mind wave sensor. That type with eye blink -Many paralyzed people tries to communicate with the surroundings but cannot. But, they can actually do it with this brain keyboard and communicate with their surroundings. Mind waves sensor reads the eye blink and accordingly text gets displayed on the text box.
3	<b>Password authentication using Brainwaves</b>	Security with Brain-As the brainwave pattern of every individual is unique. We use Brain Waves sensors in biometric identification. This can be used from high security areas to our application.
4	<b>Train your brain</b>	Neuro feedback training helps to learn to self-control the brain response with conscious feedback.
5	<b>Measuring blood pressure and heart rate to alert the patient in the case of any sudden changes</b>	We measure the heart rate per minute, through a sensor, which is installed on the patient's head.
6	<b>Measuring the level of meditation</b>	We measure the level of meditation, through a sensor, which is installed on the patient's head.

7	<b>Emotion Recognition using brainwave</b>	Emotion is purely carried by brainwaves, mind waves sensor can detect the human emotions through Brainwaves, process the data collected and classify emotions felt by respondents and reactions. When he would be sad we will record the current situation through the camera mobile to know the reason.
8	<b>Alarm patient's relatives when he starts sleep</b>	When the application detects that the patient has slept through his brain waves, which are tracked through the brain wave sensor and is also confirmed by the heart rate that reaches to the mobile phone, the relatives' application will issue an alarm.
9	<b>Calling or communicating with the patient's relative or his friends</b>	When the patient needs to communicate with his relatives or friends he can send notification for calling.
10	<b>Harmonic brain: Analyzing the brain's response to different types of music</b>	Map your brain response to different kinds of music and learn which music is the most soothing and which is the most invigorating.

**(Table 5.2) Non-Functional Requirements for System**

#	Non-functional requirements	
1	<b>Accessibility</b>	Easy design that is suited for different communities to use.
2	<b>Adaptability</b>	Suitable for different inputs like medical tests and foods.

3	<b>Availability</b>	Available for low versions of android devices as well as high versions.
4	<b>Backup</b>	Easy to recover outputs and deleted results as soft deleting rule is applied.
5	<b>Capacity</b>	Low storage space consumption.
6	<b>Minimize Response Time</b>	System response within few seconds.
7	<b>Performance</b>	High (recognition and detection) tools and algorithms.
8	<b>Reliability And Safety</b>	System should be safe so patient can rely on.
9	<b>Usability</b>	Easy to use and understand for different users and supported by consultation system (Chabot).

### 5.1.3 User Requirements

The user of the system is a patient who suffers from complete paralysis, He can move the wheelchair via brain signals, He can also write by the eye blink and he can write to his escort through the mobile application via write in it through the eye blink.

#### **Patient requirements: -**

- (1) Log in to system as (Patients/ Escort).
- (2) Moving wheelchair using mind wave sensor.
- (3) Writing expressions using eye blink.
- (4) Emergence call.
- (5) Login using brain signal.
- (6) Train Patient's brain using some of professional brain exercises.
- (7) Deliver mood condition in image form to the escort.

## **5.2 Constraints :**

### **5.2.1 Economical Constraints**

we use sensor with 3 channels instead of 5 channels because of the high cost of 5 channels sensor.

we make a small wheelchair because of the high cost of the big one .

### **5.2.2 technical Constraints**

we didn't make their part of eye detection and tracking for patient's communication because of the shortage of the time but it will be future work

### **5.2.3 logistical Constraints**

Due to the small size of our wheelchair, it cannot bear weights of more than 3 kilograms





## 7) Implementaion Details :

The implementation process is divided into two parts:

### **7.1 First Part Is The Hardware**

- 1) We made a **wheelchair**, It is considered as a core member in APP and sensor running and testing process.

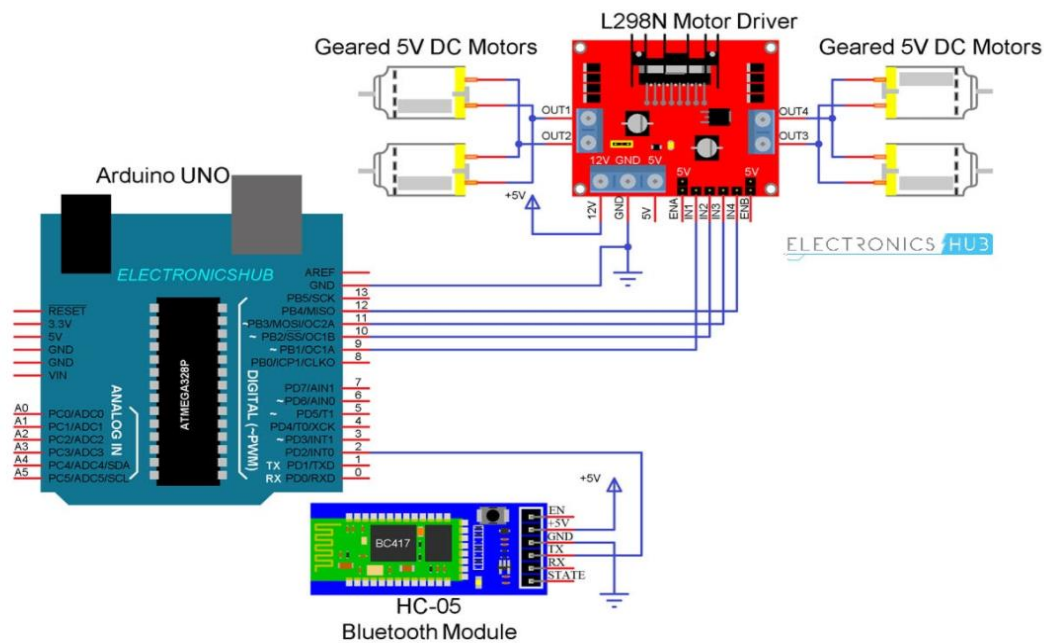


**(Fig.7.1) Wheelchair**



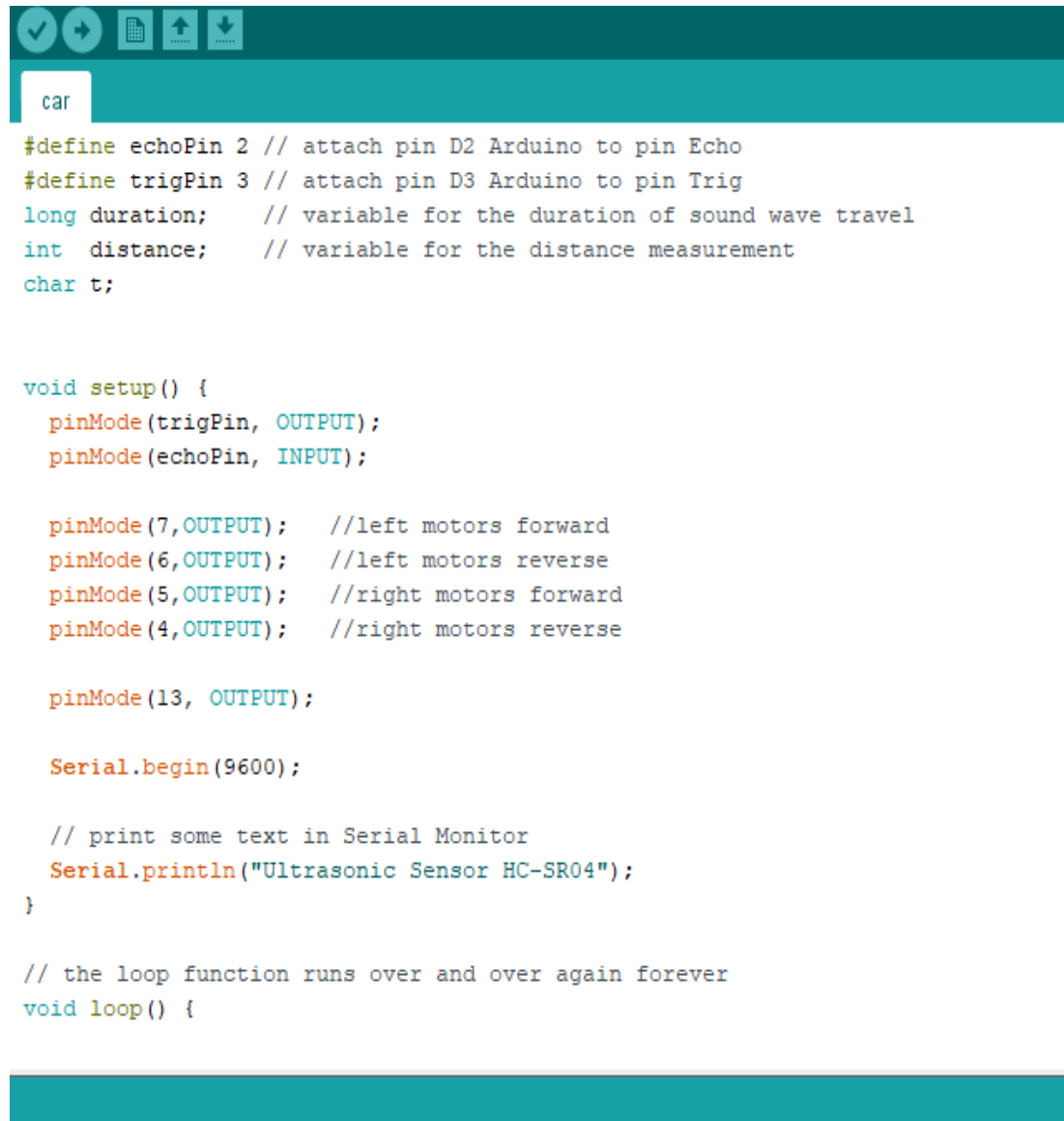
## Components required

- Arduino UNO
- L298N Motor Driver Module
- HC-05 Bluetooth Module
- Robot Chassis
- 4 x 5V Geared Motors
- Connecting Wires
- Battery Holder
- Power Supply
- Android Phone
- Bluetooth Controller App



(Fig.7.2) Circuit design

## **\*\* Code :**



```
car

#define echoPin 2 // attach pin D2 Arduino to pin Echo
#define trigPin 3 // attach pin D3 Arduino to pin Trig
long duration;    // variable for the duration of sound wave travel
int distance;     // variable for the distance measurement
char t;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  pinMode(7,OUTPUT); //left motors forward
  pinMode(6,OUTPUT); //left motors reverse
  pinMode(5,OUTPUT); //right motors forward
  pinMode(4,OUTPUT); //right motors reverse

  pinMode(13, OUTPUT);

  Serial.begin(9600);

  // print some text in Serial Monitor
  Serial.println("Ultrasonic Sensor HC-SR04");
}

// the loop function runs over and over again forever
void loop() {
```

```
car

// the loop function runs over and over again forever
void loop() {

    if(Serial.available()){
        t = Serial.read();
        Serial.println(t);
    }

    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance = duration * 0.034 / 2;
    // digitalWrite(7,LOW);
    // digitalWrite(6,LOW);
    // digitalWrite(5,HIGH);
    // digitalWrite(4,LOW);

    if(t == 'F'){          //move forward(all motors rotate in forward direction)
        digitalWrite(6,HIGH);
        digitalWrite(4,HIGH);

        digitalWrite(7,LOW);
        digitalWrite(5,LOW);
    }
}
```

```
car

    digitalWrite(13,LOW);
}
else if(t == 'B'){        //move reverse (all motors rotate in reverse direction)
    digitalWrite(7,HIGH);
    digitalWrite(5,HIGH);

    digitalWrite(6,LOW);
    digitalWrite(4,LOW);
    digitalWrite(13,LOW);
}
else if(t == 'L'){        //turn right (left side motors rotate in forward direction, right side motors doesn't rotate)
    digitalWrite(4,HIGH);

    digitalWrite(5,LOW);
    digitalWrite(6,LOW);
    digitalWrite(7,HIGH);
    digitalWrite(13,LOW);
}
else if(t == 'R'){        //turn left (right side motors rotate in forward direction, left side motors doesn't rotate)
    digitalWrite(6,HIGH);

    digitalWrite(4,LOW);
    digitalWrite(5,HIGH);
    digitalWrite(7,LOW);
    digitalWrite(13,LOW);
}
else if(t == 'S'){        //STOP (all motors stop)
```

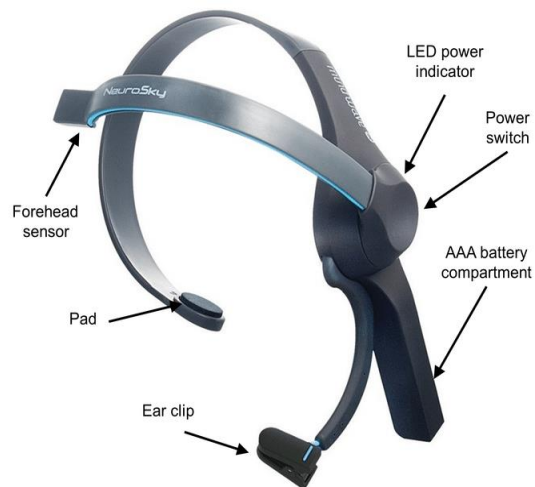
```
car
digitalWrite(4,LOW);
digitalWrite(13,LOW);
}
else if(t == 'L'){ //turn right (left side motors rotate in forward direction, right side motors doesn't rotate)
digitalWrite(4,HIGH);

digitalWrite(5,LOW);
digitalWrite(6,LOW);
digitalWrite(7,HIGH);
digitalWrite(13,LOW);
}
else if(t == 'R'){ //turn left (right side motors rotate in forward direction, left side motors doesn't rotate)
digitalWrite(6,HIGH);

digitalWrite(4,LOW);
digitalWrite(5,HIGH);
digitalWrite(7,LOW);
digitalWrite(13,LOW);
}
else if(t == 'S'){ //STOP (all motors stop)
digitalWrite(7,LOW);
digitalWrite(6,LOW);
digitalWrite(5,LOW);
digitalWrite(4,LOW);
digitalWrite(13,HIGH);
}
delay(100);
}
```

## 2) NeuroSky Brain Computer Interface

NeuroSky develops low cost easy to use ASICs for acquiring electroencephalography (EEG) from users through a wireless headset. The major advantage of the dry sensor is the time it takes to set up. Traditional gel bases EEGs can take up to 30 minutes to start acquiring data while the NeuroSky headsets are ready to go in seconds. The disadvantage to the dry sensor electrodes is an increase in the impedance contact the sensor has with the scalp. This can cause low amplitude signals to not be fully represented in the acquired data.



**(Fig.7.3) Neurosky Headset**

### **How It Works :**

Escort help patient to put The Neurosky headset. The user registers for the system for the first time; thereafter, he only logs in; however, he can register using face recognition. The second phase involves connecting the EEG sensor to the app via the system's Bluetooth module, followed by some brain training exercises to train the user's brain to think correctly. After that, the user can only move his wheelchair using his brain mind wave: The EEG sensor transmits brain waves to the app, which uses these waves as inputs to the Arduino board, which controls the motors that move the chair. The user can also type in a specific customized keyboard using only his eye blinks, which are recorded by an EEG sensor, and eventually communicate with the peripheral world. For his own safety, the user can also make an emergency call to request assistance.

## 7.2 Software.....> Second Part Is The APP:

We built an application that have two separate interfaces as we declared in GUI .

1. **Patient interface:** that enables patient to express his needs, chatting and monitoring his health status.
2. **Escort interface:** that enables patient's escort to move patients chair, chat with patient and monitor patient's health status.

### 7.2.1 APP Implementation :

#### 7.2.1.1 Code Sample for Connection

```
1  import 'dart:async';
2  import 'package:flutter/material.dart';
3  import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';
4  import 'package:servo_app/device.dart';
5
6  class SelectBondedDevicePage extends StatefulWidget {
7    /// If true, on page start there is performed discovery upon the bonded devices.
8    /// Then, if they are not available, they would be disabled from the selection.
9    final bool checkAvailability;
10   final Function onCahtPage;
11
12   const SelectBondedDevicePage(
13     {this.checkAvailability = true, @required this.onCahtPage});
14
15   @override
16   _SelectBondedDevicePage createState() => new _SelectBondedDevicePage();
17 }
18
19 enum _DeviceAvailability {
20   no,
21   maybe,
22   yes,
23 }
24
25 class _DeviceWithAvailability extends BluetoothDevice {
26   BluetoothDevice device;
27   _DeviceAvailability availability;
28   int rssi;
29
30   _DeviceWithAvailability(this.device, this.availability, [this.rssi]);
31 }
32
33 class _SelectBondedDevicePage extends State<SelectBondedDevicePage> {
34   List<_DeviceWithAvailability> devices = List<_DeviceWithAvailability>();
35
36   // Availability
37   StreamSubscription<BluetoothDiscoveryResult> _discoveryStreamSubscription;
38   bool _isDiscovering;
39
40   SelectBondedDevicePage();
```

```

39
40   _SelectBondedDevicePage();
41
42   @override
43   ▼ void initState() {
44     super.initState();
45
46     _isDiscovering = widget.checkAvailability;
47
48     ▼ if (_isDiscovering) {
49       _startDiscovery();
50     }
51
52     // Setup a list of the bonded devices
53     FlutterBluetoothSerial.instance
54       .getBondedDevices()
55     ▼ .then((List<BluetoothDevice> bondedDevices) {
56     ▼   setState(() {
57       devices = bondedDevices
58         .map(
59           (device) => _DeviceWithAvailability(
60             device,
61             widget.checkAvailability
62               ? _DeviceAvailability.maybe
63               : _DeviceAvailability.yes,
64           ),
65         )
66       .toList();
67     });
68   });
69 }
70
71 ▼ void _restartDiscovery() {
72 ▼   setState(() {
73     _isDiscovering = true;
74   });
75
76   _startDiscovery();
77 }
78
79 ▼ void _startDiscovery() {

```

```

79 ▼ void _startDiscovery() {
80     _discoveryStreamSubscription =
81     FlutterBluetoothSerial.instance.startDiscovery().listen((r) {
82     setState(() {
83         Iterator i = devices.iterator;
84         while (i.moveNext()) {
85             var _device = i.current;
86             if (_device.device == r.device) {
87                 _device.availability = _DeviceAvailability.yes;
88                 _device.rssi = r.rssi;
89             }
90         }
91     });
92 });
93
94 ▼ _discoveryStreamSubscription.onDone(() {
95     setState(() {
96         _isDiscovering = false;
97     });
98 });
99 }
100
101 @override
102 ▼ void dispose() {
103     // Avoid memory leak ('setState' after dispose) and cancel discovery
104     _discoveryStreamSubscription?.cancel();
105
106     super.dispose();
107 }
108
109 @override
110 ▼ Widget build(BuildContext context) {
111     List<BluetoothDeviceListEntry> list = devices
112         .map(
113             (_device) => BluetoothDeviceListEntry(
114                 device: _device.device,
115                 // rssi: _device.rssi,
116                 // enabled: _device.availability == _DeviceAvailability.yes,
117             onTap: () {
118                 widget.onCahtPage(_device.device);

```

**(Fig. 7.4) Code Sample for Connection**



### 7.2.1.2 Code sample for connected devices

```
1 import 'package:flutter/material.dart';
2 import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';
3
4 class BluetoothDeviceListEntry extends StatelessWidget {
5   final Function onTap;
6   final BluetoothDevice device;
7
8   BluetoothDeviceListEntry({this.onTap, @required this.device});
9
10  @override
11  Widget build(BuildContext context) {
12    return ListTile(
13      onTap: onTap,
14      leading: Icon(Icons.devices),
15      title: Text(device.name ?? "Unknown device"),
16      subtitle: Text(device.address.toString()),
17      trailing: FlatButton(
18        child: Text('Connect'),
19        onPressed: onTap,
20        color: Colors.greenAccent,
21      ),
22    );
23  }
24 }
25 |
```

(Fig.7.5) Code Sample for Connection

### 7.2.1.3 Connecting EEG sensor to the System

```
90     }
91     });
92     });
93
94     _discoveryStreamSubscription.onDone(() {
95         setState(() {
96             _isDiscovering = false;
97         });
98     });
99 }
100
101 @override
102 void dispose() {
103     // Avoid memory leak (`setState` after dispose) and cancel discovery
104     _discoveryStreamSubscription?.cancel();
105
106     super.dispose();
107 }
108
109 @override
110 Widget build(BuildContext context) {
111     List<BluetoothDeviceListEntry> list = devices
112         .map(
113             (_device) => BluetoothDeviceListEntry(
114                 device: _device.device,
115                 // rssi: _device.rssi,
116                 // enabled: _device.availability == _DeviceAvailability.yes,
117                 onTap: () {
118                     widget.onCahtPage(_device.device);
119                 },
120             ),
121         )
122         .toList();
123     return ListView(
124         children: list,
125     );
126 }
127 }
128 }
```

```
import mindwave, time, csv
# Creating a new Headset object to establish the connection with dongle.
# pass the Port ==> COM4 / pass the headset ID ==> FD48 Notes:- You can find it in the sensor's battery holder.
headset = mindwave.Headset('COM4', 'FD48')
# Recommended to wait at least a couple seconds before connecting the dongle to the headset.
time.sleep(2)
# Connecting with the mindwave sensor.
headset.connect()
while headset.status != 'connected':
    time.sleep(2)
    if headset.status == 'standby':
        headset.connect()
print("Connected.")
# Creating 3 columns (Attention, Meditation, Blink) for csv file.
header = ['attention', 'meditation', 'blink']
# Creating empty list to append the fetched data from mindwave sensor.
data = []
# Opening the file in the write mode and encoding UTF8.
file = open('mindwave_signal', 'w', encoding='UTF8')
# Creating the csv writer.
writer = csv.writer(file)
# Writing the header.
writer.writerow(header)
while True:
    # Append data to the list.
    data.append(headset.attention)
    data.append(headset.meditation)
    data.append(headset.blink)
    # Writing the data.
    writer.writerow(data)
file.close()
```

(Fig.7.6) Connecting EEG sensor

## Output for previous code:

	A	B	C
1	attention	meditation	blink
2	44.531	38.477	24.316
3	21.582	36.816	27.148
4	18.848	45.117	50.879
5	39.551	57.52	47.852
6	61.426	60.059	51.27
7	52.539	64.258	45.312
8	63.086	59.18	30.957
9	56.738	51.367	22.754
10	47.656	46.387	28.223
11	39.941	45.605	26.074
12	41.016	28.223	10.84
13	35.059	34.863	20.117
14	20.215	44.238	43.652
15	24.805	51.758	36.035
16	43.066	54.492	38.867
17	43.75	69.141	50.195
18	60.059	58.984	42.285
19	56.445	41.504	26.855
20	59.082	32.227	19.434
21	39.746	41.895	17.187
22	41.211	29.004	18.945
23	32.812	24.414	26.66

(Fig. 7.7) Output Connecting EEG Sensor

## 7.2.2 GUI Implementation

### 7.2.2.1 GUI implementation for patient or escort screen:

```
1  import 'package:flutter/material.dart';
2  import 'package:life_again/layout/bluetooth_alert.dart';
3  import 'package:life_again/layout/start_containers.dart';
4
5  class PatientOrEscort extends StatefulWidget {
6    @override
7    _PatientOrEscortState createState() => _PatientOrEscortState();
8  }
9
10 class _PatientOrEscortState extends State<PatientOrEscort> {
11   @override
12   Widget build(BuildContext context) {
13     return Scaffold(
14       body: SizedBox(
15         width: MediaQuery.of(context).size.width,
16         child: Padding(
17           padding: const EdgeInsets.symmetric(horizontal: 20.0),
18           child: Column(
19             crossAxisAlignment: CrossAxisAlignment.center,
20             mainAxisAlignment: MainAxisAlignment.spaceEvenly,
21             children: [
22               Text(
23                 'Now you can back to life and practice daily life easily',
24                 style: TextStyle(
25                   fontSize: 17.0,
26                   color: Color(0xffA2A3A6),
27                 ),
28             ),
29             Column(
30               children: [
31                 InkWell(
32                   onTap: () {
33                     showDialog(
34                       context: context,
35                       builder: (context) => AlertttDialog(),
36                     );
37                 },
38                 child: StartContainers(
39                   imgText: 'assets/images/Flat.png', text: 'Patient')),
40               SizedBox(
41                 height: 60,
42               ),
43               InkWell(
44                 onTap: () {
45                   //
46                 },
47                 child: StartContainers(
48                   imgText: 'assets/images/old-man.png', text: 'Escort'),
49               ),
50             ],
51           ),
52         ),
53       ),
54     ),
55   );
56 }
57 }
58 }
59 }
```

(Fig.7.8) GUI Implementation for Patient or Escort Screen

### 7.2.2.2 GUI implementation for Bluetooth connecting screen:

```
1 |import 'package:flutter/material.dart';
2 |import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';
3 |import 'package:life_again/modules/start_training/start_training.dart';
4
5 ▼ class AlerttDialog extends StatelessWidget {
6 ▼   AlerttDialog({
7     this.content,
8     this.shape,
9   });
10
11   var shape;
12   var content;
13
14   @override
15 ▼   Widget build(BuildContext context) {
16     return AlertDialog(
17       shape: RoundedRectangleBorder(
18         borderRadius: BorderRadius.all(Radius.circular(20.0)),),),
19       content: Container(
20         width: MediaQuery.of(context).size.width,
21         height: MediaQuery.of(context).size.height / 2,
22         decoration: new BoxDecoration(
23           shape: BoxShape.rectangle,
24           color: const Color(0xFFFFFFFF),
25           borderRadius: BorderRadius.all(Radius.circular(20.0)),
26         ),
27       child: Column(
28         mainAxisAlignment: MainAxisAlignment.spaceEvenly,
29 ▼       children: [
30         Flexible(
31           fit: FlexFit.loose,
32           child: Image(
33             image: AssetImage('assets/images/bluetooth.png'),
34           ),
35         ),
36         Text(
37           'Connect EEG Sensor with bluetooth',
38           style: TextStyle(fontWeight: FontWeight.w500),
39         ),
40         InkWell(
41 ▼       onTap: () {
42         Navigator.pushReplacement(
43           context,
44           MaterialPageRoute(
45             builder: (context) => FutureBuilder(
46               future: FlutterBluetoothSerial.instance.requestEnable(),
47               builder: (context, future) {
48 ▼             if (future.connectionState == ConnectionState.waiting) {
49               return Scaffold(
50                 body: Container(
51                   height: double.infinity,
52                   child: Center(
53                     child: Icon(
54                       Icons.bluetooth_disabled,
55                       size: 200.0,
56                       color: Colors.black12
57                     ),
58                   ),
59                 ),
60               );
61             } else {
62 ▼               return StartTraining();
63             }
64           },
65         ),
66       ),
67     );
68   },
69   child: Container(
70     width: MediaQuery.of(context).size.width,
71     height: MediaQuery.of(context).size.height / 18,
```

```

73         decoration: BoxDecoration(
74             color: Color(0xff0D8EBC),
75             borderRadius: BorderRadius.circular(10.0)),
76         child: Center(
77             child: Text(
78                 'go to Bluetooth',
79                 style: TextStyle(
80                     color: Colors.white, fontWeight: FontWeight.w500),
81             ),

```

(Fig. 7.9) GUI Implementation for Bluetooth Connecting Screen

### 7.2.2.3 GUI implementation for start brain training screen:

```

1  import 'package:flutter/material.dart';
2  import 'package:life_again/modules/brain_training/brain_training_forward.dart';
3
4  class StartTraining extends StatefulWidget {
5      @override
6      _StartTrainingState createState() => _StartTrainingState();
7  }
8
9  class _StartTrainingState extends State<StartTraining> {
10     @override
11     Widget build(BuildContext context) {
12         return Scaffold(
13             body: Padding(
14                 padding: const EdgeInsets.symmetric(vertical: 40, horizontal: 20),
15                 child: Column(
16                     crossAxisAlignment: CrossAxisAlignment.start,
17                     mainAxisAlignment: MainAxisAlignment.spaceEvenly,
18                     children: [
19                         Flexible(
20                             child: InkWell(
21                                 onTap: () {
22                                     Navigator.of(context).pop();
23                                 },
24                                 child: Container(
25                                     margin: EdgeInsets.only(
26                                         right: 15.0, top: 15.0, bottom: 15.0),
27                                     width: 30.0,
28                                     height: 30.0,
29                                     decoration: BoxDecoration(
30                                         image: DecorationImage(
31                                             image: AssetImage(
32                                                 "assets/images/back.png")),
33                                     ),
34                                 ),
35                         ),
36                         SizedBox(
37                             height: MediaQuery.of(context).size.height - 200,
38                             child: Column(
39                                 crossAxisAlignment: CrossAxisAlignment.center,

```

```

40      mainAxisAlignment: MainAxisAlignment.spaceBetween,
41      children: [
42        Padding(
43          padding: const EdgeInsets.only(top: 40),
44          child: Container(
45            child: Column(
46              mainAxisAlignment: MainAxisAlignment.center,
47              children: [
48                RichText(
49                  textAlign: TextAlign.center,
50                  text: TextSpan(children: <TextSpan>[
51                    TextSpan(
52                      text:
53                        "After connect EEG sensor with Bluetooth device now you should test it By doing a ",
54                      style: TextStyle(
55                        color: Colors.black87,
56                        fontWeight: FontWeight.w500,
57                        fontSize: 20)),
58                    TextSpan(
59                      text: "brain training",
60                      style: TextStyle(
61                        color: Color(0xff0D8EBC),
62                        fontWeight: FontWeight.w500,
63                        fontSize: 20)),
64                  ]),
65                ),
66              ],
67            ),
68          ),
69        ),
70        Flexible(
71          fit: FlexFit.loose,
72          child: Image.asset('assets/images/start.png'),
73        ),
74        InkWell(
75          onTap: () {
76            Navigator.of(context).push(MaterialPageRoute(
77              builder: (context) => BrainTrainingForward());
78            ),
79          child: Container(
80            decoration: BoxDecoration(
81              color: Color(0xff0D8EBC),
82              borderRadius: BorderRadius.circular(10.0)),
83            width: MediaQuery.of(context).size.width,
84            height: MediaQuery.of(context).size.height / 15,
85            child: Center(
86              child: Text(
87                'Start training',
88                style: TextStyle(
89                  fontSize: 25,
90                  fontWeight: FontWeight.w600,
91                  color: Colors.white),

```

(Fig. 7.10) GUI Implementation for Start Brain Training Screen

### 7.2.2.4 GUI implementation for forward moving screen:

```
1 import 'package:flutter/material.dart';
2 import 'package:life_again/modules/brain_training/brain_training_back.dart';
3 import 'package:life_again/shared/components/brain_training/Button.dart';
4 import 'package:life_again/shared/components/brain_training/instruction.dart';
5 import 'package:life_again/shared/constants/brain_training_title.dart';
6 import 'package:life_again/shared/components/brain_training/directions.dart';
7 import 'package:life_again/shared/components/brain_training/progress.dart';
8
9 class BrainTrainingForward extends StatefulWidget {
10   @override
11   _BrainTrainingForwardState createState() => _BrainTrainingForwardState();
12 }
13
14 class _BrainTrainingForwardState extends State<BrainTrainingForward> {
15   @override
16   Widget build(BuildContext context) {
17     return Scaffold(
18       body: SafeArea(
19         child: Padding(
20           padding: const EdgeInsets.symmetric(horizontal: 20.0),
21           child: Stack(
22             children: [
23               SingleChildScrollView(
24                 child: Column(
25                   children: [
26                     brainTrainingTitle(context, "Brain Training"),
27                     SizedBox(
28                       height: MediaQuery.of(context).size.height * 0.02,
29                     ),
30                     progress(context, 0.25),
31                     SizedBox(
32                       height: MediaQuery.of(context).size.height * 0.09,
33                     ),
34                     singleArrow(
35                       context, "assets/images/forwardArrowFilled.png"),
36                     SizedBox(
37                       height: MediaQuery.of(context).size.height * 0.02,
38                     ),
39                     middleRow(context, "assets/images/leftArrow.png",
40                               "assets/images/rightArrow.png"),
41                     SizedBox(
42                       height: MediaQuery.of(context).size.height * 0.02,
43                     ),
44                     singleArrow(context, "assets/images/downArrow.png"),
45                     SizedBox(
46                       height: MediaQuery.of(context).size.height * 0.07,
47                     ),
48                     instruction("Forward"),
49                   ],
50                 ),
51               ),
52             ],
53             sideButton("Next", () {
54               Navigator.of(context).push(
55                 MaterialPageRoute(builder: (context) => BrainTrainingBack()),
56               );
57             }, Alignment.bottomRight),
58           ),
59         ),
60       ),
61     );
62 }
```

(Fig. 7.11) GUI Implementation for Forward Moving Screen



### 7.2.2.5 GUI implementation for backward moving screen:

```
1 import 'package:flutter/material.dart';
2 import 'package:life_again/modules/brain_training/brain_training_right.dart';
3 import 'package:life_again/shared/components/brain_training/Button.dart';
4 import 'package:life_again/shared/components/brain_training/directions.dart';
5 import 'package:life_again/shared/components/brain_training/instruction.dart';
6 import 'package:life_again/shared/components/brain_training/progress.dart';
7 import 'package:life_again/shared/constants/brain_training_title.dart';
8
9 class BrainTrainingBack extends StatefulWidget {
10   @override
11   _BrainTrainingBackState createState() => _BrainTrainingBackState();
12 }
13
14 class _BrainTrainingBackState extends State<BrainTrainingBack> {
15   @override
16   Widget build(BuildContext context) {
17     return Scaffold(
18       body: SafeArea(
19         child: Padding(
20           padding: const EdgeInsets.symmetric(horizontal: 20.0),
21           child: Stack(
22             children: [
23               SingleChildScrollView(
24                 child: Column(
25                   children: [
26                     brainTrainingTitle(context, "Brain Training"),
27                     SizedBox(
28                       height: MediaQuery.of(context).size.height * 0.02,
29                     ),
30                     progress(context, 0.5),
31                     SizedBox(
32                       height: MediaQuery.of(context).size.height * 0.09,
33                     ),
34                     singleArrow(context, "assets/images/forwardArrow.png"),
35                     SizedBox(
36                       height: MediaQuery.of(context).size.height * 0.02,
37                     ),
38                     middleRow(context, "assets/images/leftArrow.png",
39                       "assets/images/rightArrow.png"),
40                     SizedBox(
41                       height: MediaQuery.of(context).size.height * 0.02,
42                     ),
43                     singleArrow(context, "assets/images/downArrowFilled.png"),
44                     SizedBox(
45                       height: MediaQuery.of(context).size.height * 0.07,
46                     ),
47                     instruction("Back"),
48                   ],
49                 ),
50               ),
51               Row(
52                 mainAxisAlignment: MainAxisAlignment.spaceBetween,
53                 children: [
54                   sideButton("Back", () { Navigator.of(context).pop(); }, Alignment.bottomLeft),
55                   sideButton("Next", () {
56                     Navigator.of(context).push(MaterialPageRoute(
57                       builder: (context) => BrainTrainingRight()));
58                   }, Alignment.bottomRight),
59                 ],
60               ),
61             ],
62           ),
63         ));
64   }
65 }
```

(Fig. 7.12) GUI Implementation for Backward Moving Screen

#### 7.2.2.6 GUI implementation for right direction moving screen:

```
1 import 'package:flutter/material.dart';
2 import 'package:life_again/modules/brain_training/brain_training_left.dart';
3 import 'package:life_again/shared/components/brain_training/Button.dart';
4 import 'package:life_again/shared/components/brain_training/Instruction.dart';
5 import 'package:life_again/shared/constants/brain_training_title.dart';
6 import 'package:life_again/shared/components/brain_training/directions.dart';
7 import 'package:life_again/shared/components/brain_training/progress.dart';
8
9 class BrainTrainingRight extends StatefulWidget {
10   @override
11   _BrainTrainingRightState createState() => _BrainTrainingRightState();
12 }
13
14 class _BrainTrainingRightState extends State<BrainTrainingRight> {
15   @override
16   Widget build(BuildContext context) {
17     return Scaffold(
18       body: SafeArea(
19         child: Padding(
20           padding: const EdgeInsets.symmetric(horizontal: 20.0),
21           child: Stack(
22             children: [
23               SingleChildScrollView(
24                 child: Column(
25                   children: [
26                     brainTrainingTitle(context, "Brain Training"),
27                     SizedBox(
28                       height: MediaQuery.of(context).size.height * 0.02,
29                     ),
30                     progress(context, 0.75),
31                     SizedBox(
32                       height: MediaQuery.of(context).size.height * 0.09,
33                     ),
34                     singleArrow(context, "assets/images/forwardArrow.png"),
35                     SizedBox(
36                       height: MediaQuery.of(context).size.height * 0.02,
37                     ),
38                     middleRow(context, "assets/images/leftArrow.png",
39                       "assets/images/rightArrowFilled.png"),
40                     SizedBox(
41                       height: MediaQuery.of(context).size.height * 0.07,
42                     ),
43                     instruction("Right"),
44                   ],
45                 ),
46             ),
47           ),
48         ),
49       ),
50       Row(
51         mainAxisAlignment: MainAxisAlignment.spaceBetween,
52         children: [
53           sideButton("Back", () { Navigator.of(context).pop(); }, Alignment.bottomLeft ),
54           sideButton("Next", () {
55             Navigator.of(context).push(MaterialPageRoute(
56               builder: (context) => BrainTrainingLeft());
57             }, Alignment.bottomRight),
58         ],
59       ),
60     );
61   }
62 }
```

(Fig. 7.13) GUI Implementation for Right Direction Moving Screen

### 7.2.2.7 GUI implementation for left direction moving screen:

```
1 import 'package:flutter/material.dart';
2 import 'package:life_again/modules/patient_information/patient_information.dart';
3 import 'package:life_again/shared/components/brain_training/Button.dart';
4 import 'package:life_again/shared/components/brain_training/instruction.dart';
5 import 'package:life_again/shared/constants/brain_training_title.dart';
6 import 'package:life_again/shared/components/brain_training/directions.dart';
7 import 'package:life_again/shared/components/brain_training/progress.dart';
8
9 class BrainTrainingLeft extends StatefulWidget {
10   @override
11   _BrainTrainingLeftState createState() => _BrainTrainingLeftState();
12 }
13
14 class _BrainTrainingLeftState extends State<BrainTrainingLeft> {
15   @override
16   Widget build(BuildContext context) {
17     return Scaffold(
18       body: SafeArea(
19         child: Padding(
20           padding: const EdgeInsets.symmetric(horizontal: 20.0),
21           child: Stack(
22             children: [
23               SingleChildScrollView(
24                 child: Column(
25                   children: [
26                     brainTrainingTitle(context, "Brain Training"),
27                     SizedBox(
28                       height: MediaQuery.of(context).size.height * 0.02,
29                     ),
30                     progress(context, 1.0),
31                     SizedBox(
32                       height: MediaQuery.of(context).size.height * 0.09,
33                     ),
34                     singleArrow(context, "assets/images/forwardArrow.png"),
35                     SizedBox(
36                       height: MediaQuery.of(context).size.height * 0.02,
37                     ),
38                     middleRow(context, "assets/images/leftArrowFilled.png",
39                       "assets/images/rightArrow.png"),
40                     SizedBox(
41                       height: MediaQuery.of(context).size.height * 0.02,
42                     ),
43                     singleArrow(context, "assets/images/downArrow.png"),
44                     SizedBox(
45                       height: MediaQuery.of(context).size.height * 0.07,
46                     ),
47                     instruction("Left"),
48                   ],
49                 ),
50               Row(
51                 mainAxisAlignment: MainAxisAlignment.spaceBetween,
52                 children: [
53                   sideButton("Back", () { Navigator.of(context).pop(); }, Alignment.bottomLeft),
54                   sideButton("Finish", () {
55                     Navigator.of(context).push(MaterialPageRoute(
56                       builder: (context) => PatientInformation()));
57                   }, Alignment.bottomRight),
58                 ],
59               ),
60             ],
61           ),
62         ),
63       ),
64     );
65   }
66 }
```

(Fig. 7.14) GUI Implementation for Left Direction Moving Screen

### 7.2.2.8 GUI implementation for patient information screen:

```
1 import 'package:flutter/material.dart';
2 import 'package:life_again/modules/patient/patient_home.dart';
3 import 'package:life_again/shared/components/brain_training/Button.dart';
4 import 'package:life_again/shared/components/patient_information/alertDialog.dart';
5 import 'package:life_again/shared/components/patient_information/information_field.dart';
6 import 'package:life_again/shared/constants/brain_training_title.dart';
7
8 class PatientInformation extends StatefulWidget {
9   @override
10   _PatientInformationState createState() => _PatientInformationState();
11 }
12
13 class _PatientInformationState extends State<PatientInformation> {
14   String gender = "Select";
15   String gender1 = "Male";
16   String gender2 = "Female";
17   int age = 22;
18   int weight = 80;
19
20   @override
21   Widget build(BuildContext context) {
22     return Scaffold(
23       body: SafeArea(
24         child: Padding(
25           padding: const EdgeInsets.symmetric(horizontal: 20.0),
26           child: Stack(
27             children: [
28               SingleChildScrollView(
29                 child: Column(
30                   children: [
31                     brainTrainingTitle(context, "Tell us about your patient"),
32                     SizedBox(
33                       height: MediaQuery.of(context).size.height * 0.02,
34                     ),
35                     Padding(
36                       padding: const EdgeInsets.only(left: 8.0),
37                       child: Container(
38                         width: MediaQuery.of(context).size.width,
39                         height: MediaQuery.of(context).size.height * 0.06,
40                         child: Text(
41                           "Your individual parameters are important for monitoring your patient health with you.",
42                           style: TextStyle(
43                             fontSize: 17.0,
44                             color: Color(0xFFA2A3A6),
45                           ),
46                         ),
47                       ),
48                     ),
49                     SizedBox(
50                       height: MediaQuery.of(context).size.height * 0.05,
51                     ),
52                     InformationField(
53                       context,
54                       "assets/images/user.png",
55                       "Patient gender",
56                       gender,
57                       () {
58                         showDialog(
59                           context: context,
60                           builder: (context) {
61                             return alertGender(context, () {
62                               setState(() {
63                                 gender = gender1;
64                               });
65                             }, () {
66                               setState(() {
67                                 gender = gender2;
68                               });
69                             });
70                           },
71                         ),
72                       ),
73                     ),
74                     InformationField(
75                       context,
76                       "assets/images/weighing-scale.png",
77                       "Patient weight",
78                       "${weight} kg",
```

```

77         Patient weight",
78         "${weight} kg",
79         () {},
80         Slider(
81             value: weight.toDouble(),
82             max: 250,
83             onChanged: (double value) {
84                 setState(() {
85                     weight = value.round();
86                 });
87             },
88             activeColor: Color.fromRGBO(51, 47, 47, 1)
89         ),
90     ),
91     informationField(
92         context,
93         "assets/images/cake.png",
94         "Patient age",
95         "${age}",
96         () {},
97         Slider(
98             value: age.toDouble(),
99             max: 150,
100             onChanged: (double value) {
101                 setState(() {
102                     age = value.round();
103                 });
104             },
105             activeColor: Color.fromRGBO(51, 47, 47, 1)
106         ),
107     ),
108     informationField(
109         context,
110         "assets/images/qr.png",
111         "Patient code",
112         "Enter",
113         "Enter",
114         () {},
115         SizedBox(height: 0.0,)),
116     ],
117 ),
118 ),
119 mainButton(context, "Next", () {
120     Navigator.of(context).push(MaterialPageRoute(builder: (context) => PatientHome()));
121 })
122 ],
123 ),
124 ),
125 ),
126 );
127 }
128 }
129

```

(Fig.7.15) GUI Implementation for Patient Information Screen

### 7.2.2.9 GUI implementation for patient home page screen:

```
1 import 'package:flutter/material.dart';
2 import 'package:life_again/layout/bottomnavbar.dart';
3 import 'package:life_again/layout/patient_home_containers.dart';
4
5 class PatientHome extends StatefulWidget {
6   @override
7   _PatientHomeState createState() => _PatientHomeState();
8 }
9
10 class _PatientHomeState extends State<PatientHome> {
11   @override
12   Widget build(BuildContext context) {
13     return Scaffold(
14       backgroundColor: Color(0xffE3EAE5),
15       body: Padding(
16         padding: const EdgeInsets.only(top: 30, left: 15, right: 15),
17         child: SingleChildScrollView(
18           child: Column(
19             children: [
20               Padding(
21                 padding: const EdgeInsets.all(5),
22                 child: Row(
23                   mainAxisAlignment: MainAxisAlignment.spaceBetween,
24                   children: [
25                     Text(
26                       'Patient',
27                       style: TextStyle(
28                         fontSize: 30,
29                         fontWeight: FontWeight.w500,
30                         color: Colors.black),
31                     ),
32                     InkWell(
33                       onTap: () {
34                         Navigator.pop(context);
35                       },
36                       child: Container(
37                         width: 50,
38                         height: 50,
39                         decoration: BoxDecoration(
40                           borderRadius: BorderRadius.circular(50),
41                           color: Colors.white,
42                           border: Border.all(
43                             width: 0.5, color: const Color(0xFF08EBC)),
44                           child: Image(
45                             image: AssetImage('assets/images/roberto.png'),
46                           ),
47                         ),
48                     ),
49                   ],
50                 ),
51               ),
52               SizedBox(
53                 height: MediaQuery.of(context).size.height / 20,
54               ),
55               InkWell(
56                 onTap: () {},
57                 child: CardContainer(
58                   imgText: 'assets/images/download.png', text: 'Need food'),
59               ),
60               InkWell(
61                 onTap: () {},
62                 child: CardContainer(
63                   imgText:
64                     'assets/images/PikPng.com_glass-of-water-png_418080.png',
65                   text: 'Need Water'),
66               ),
67               InkWell(
68                 onTap: () {},
69                 child: CardContainer(
70                   imgText: 'assets/images/toilet.png', text: 'Bathroom'),
71               ),
72               InkWell(
73                 onTap: () {},
74                 child: CardContainer(
```

```

74         child: CardContainer(
75           imgText: 'assets/images/PngItem_349739.png',
76           text: 'Chat With Escort'),
77       ),
78       InkWell(
79         onTap: () {},
80         child: CardContainer(
81           imgText: 'assets/images/pencil.png', text: 'Notes'),
82       ),
83     ],
84   ),
85 ),
86 ),
87 bottomNavigationBar: BottomNavBar(),
88 );

```

(Fig. 7.16) GUI Implementation for Patient Home Page Screen

#### 7.2.2.10 GUI implementation for escort home page screen:

```

1  import 'package:flutter/material.dart';
2  import 'gridview_containers.dart';
3  import 'bottomnavbar.dart';
4
5  class UserProfile extends StatefulWidget {
6    @override
7    _UserProfileState createState() => _UserProfileState();
8  }
9
10 class _UserProfileState extends State<UserProfile> {
11
12
13   @override
14   Widget build(BuildContext context) {
15     return Scaffold(
16       backgroundColor: Color(0xFFFFFFF2),
17       body: Column(
18         mainAxisAlignment: MainAxisAlignment.center,
19         children: [
20           Padding(
21             padding: const EdgeInsets.symmetric(horizontal: 10, vertical: 40),
22             child: Row(
23               crossAxisAlignment: CrossAxisAlignment.start,
24               mainAxisAlignment: MainAxisAlignment.spaceBetween,
25               children: [
26                 IconButton(
27                   icon: Icon(Icons.arrow_back_ios),
28                   color: Color(0xFFFFD454),
29                   iconSize: 30,
30                   onPressed: () {
31                     Navigator.pop(context);
32                   },
33                 ),
34                 Container(
35                   width: MediaQuery.of(context).size.width / 3,
36                   height: MediaQuery.of(context).size.height / 6,
37                   decoration: BoxDecoration(
38                     borderRadius: BorderRadius.circular(100),
39                   ),
40                   child: Image(

```

```

41         image: AssetImage('assets/images/roberto1.png'),
42     ),
43 ),
44     IconButton(
45       icon: Icon(Icons.more_vert),
46       color: Color(0xFFFD454),
47       iconSize: 30,
48       onPressed: () {},
49     )
50   ],
51 ),
52 ),
53   Text(
54     'Mariam Abd El-Aty',
55     style: TextStyle(fontWeight: FontWeight.w900, fontSize: 22),
56   ),
57   SizedBox(
58     height: 10,
59   ),
60   Container(
61     width: MediaQuery.of(context).size.width / 4,
62     height: 2,
63     color: Color(0xFFFD454),
64   ),
65   Flexible(
66     child: Padding(
67       padding: const EdgeInsets.only(top: 20),
68       child: Container(
69         child: GridView.count(
70           primary: false,
71           padding: const EdgeInsets.all(20),
72           crossAxisSpacing: 40,
73           mainAxisSpacing: 40,
74           crossAxisCount: 2,
75           children: [
76             GridContainers(
77               imgText: 'assets/images/wheelchair.png',
78               text: 'Chair Movements',

```

```

79             ),
80             GridContainers(
81               imgText: 'assets/images/brain.png',
82               text: 'Chair Movements',
83             ),
84             GridContainers(
85               imgText: 'assets/images/wheelchair.png',
86               text: 'Chair Movements',
87             ),
88             GridContainers(
89               imgText: 'assets/images/wheelchair.png',
90               text: 'Chair Movements',
91             )
92           ],
93         ),
94       ),
95     ),
96   ),
97 ],
98 ),
99   bottomNavigationBar: BottomNavBar(),
100 );

```

(Fig. 7.17) GUI Implementation for Escort Home Page Screen



### 7.3 Algorithmic components

Before buying EEG sensor we use Machine learning technique for making a model as a simulation of sensor .we use one of classification algorithms which named Support Vector Machine (SVM) for training our model .

**Data Set:**

we use a dataset of EEG for human which contain 40000 records and we take the signal from 19 channels

**Output :**

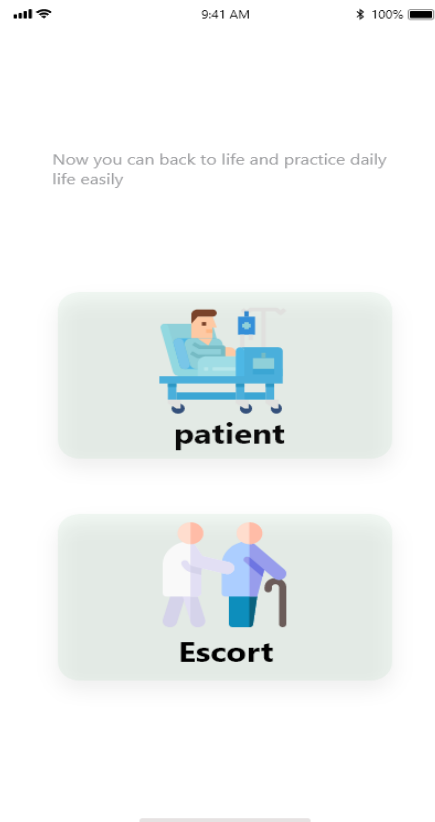
The output is the direction of input signal .

### 7.4 User Interfaces :

#### 7.4.1 Patient Interface

First time, the user signup for the system, later he only login, however, he can register using face recognition technique, The second phase is to connect the EEG sensor with the app using system's Bluetooth module, Then the patient will do some brain training exercises to train his brain to think correctly, After that the patient can move his wheelchair using his brain thoughts only, Patient can also type his thoughts and chat with his escorts with a specific customized keyboard using only his eye blink which is recorded by EEG sensor for his safety, patient can also select any item using his eye blink, access his health information, order any technical support and managing his profile.

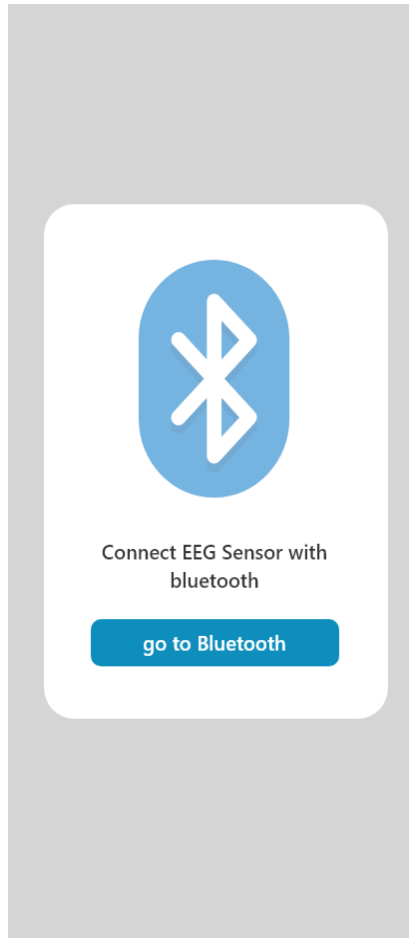
## 1) User choose if he patient or escort



(Fig.7.18) Choose Patient or Escort

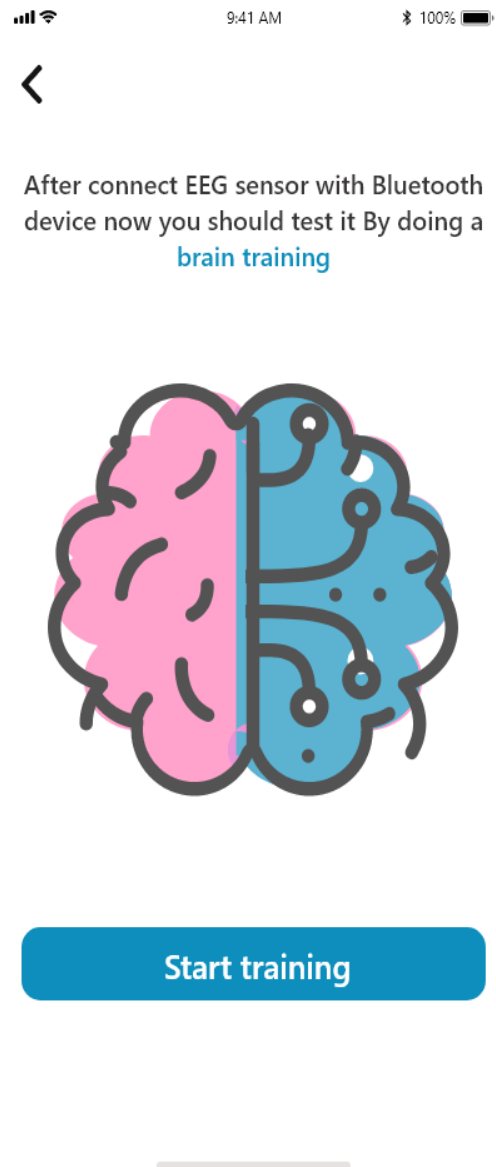
**“IF user is patient”**

**2) Start connect EEG sensor with Bluetooth:**



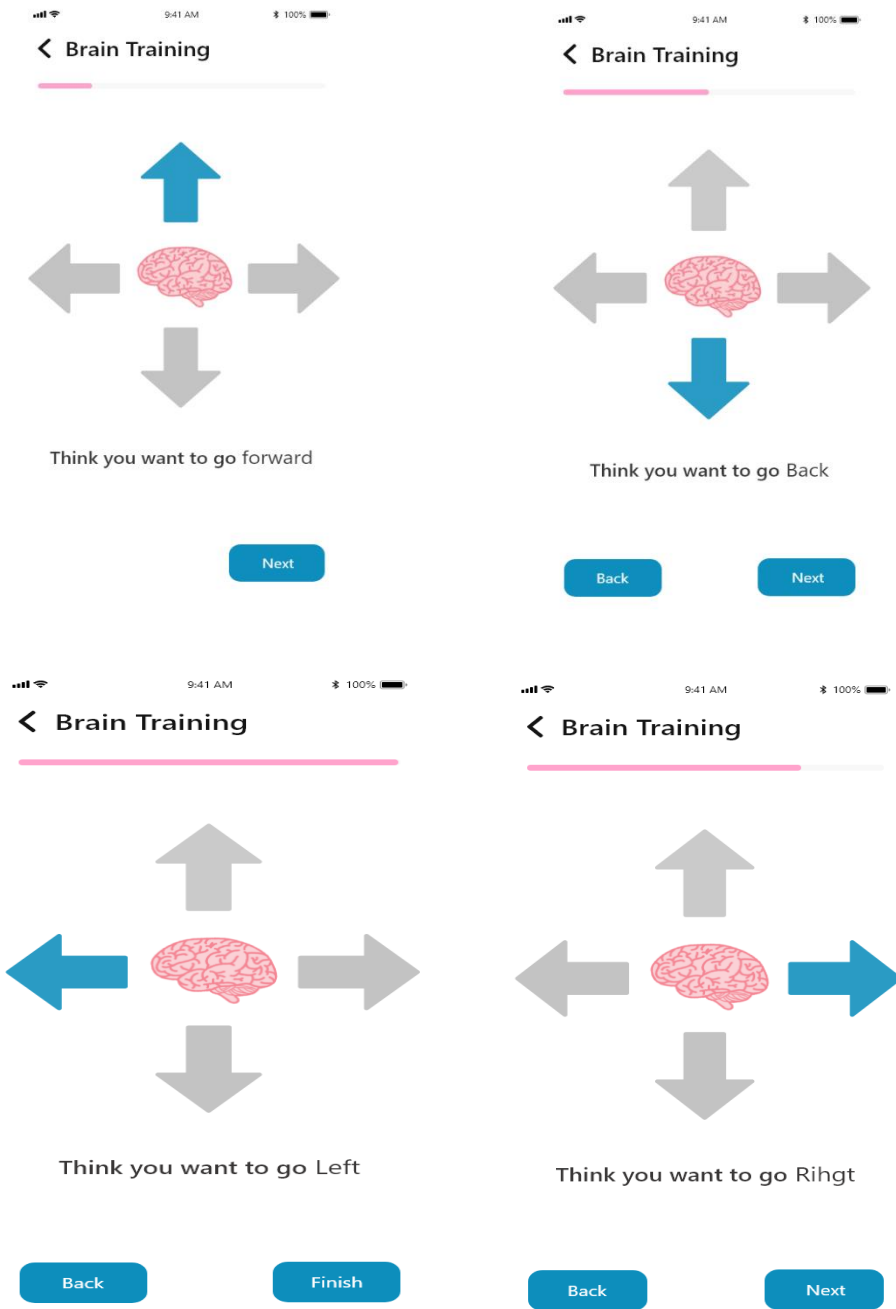
**(Fig.7.19) Connect EEG Sensor with Bluetooth**

### 3) Start training:



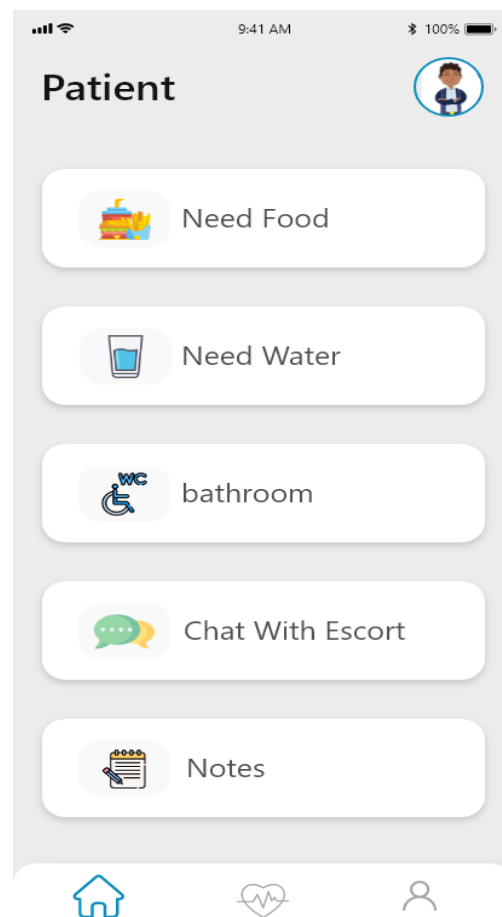
(Fig.7.20) Training

#### 4) Think about direction:



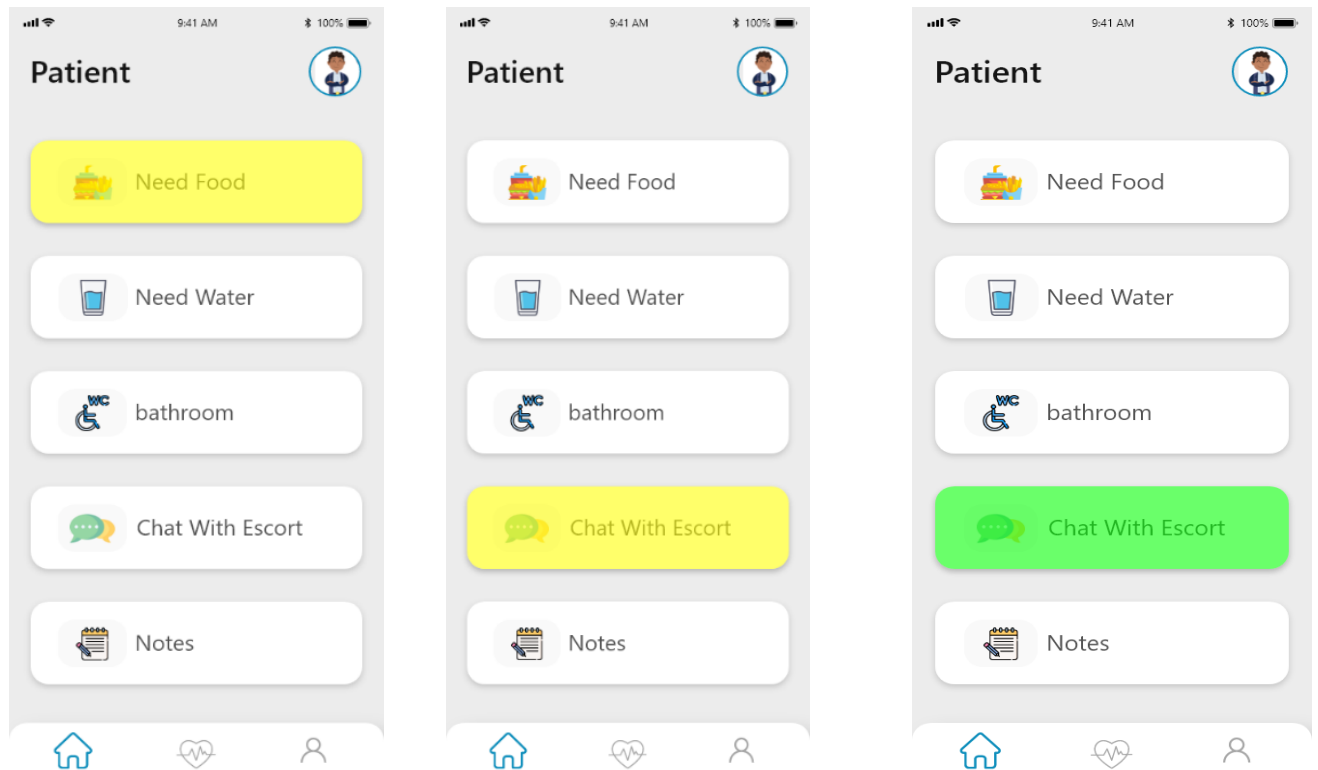
(Fig.7.21) Directions

**5) There are some options to facilitate the patient's needs**



**(Fig.7.22) Patient Needs**

## 6) Select need by blink

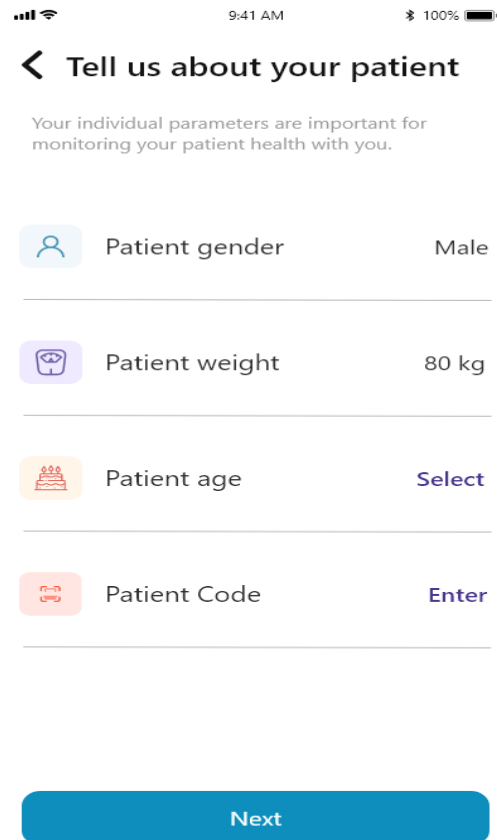


(Fig.7.23 ) Select Need





### 7.4.2 Patient Escort Interface

After sign up or log in as an escort, the escort can access and manage patient's data by (patient's id), the escort can move the chair by using the APP, and can follow-up patient's health condition and mood which are recorded by EEG sensor and stored in simple EHR system attached to APP. Escorts can also chat with patient remotely or access patient's chat, order any technical support and managing his profile.

## 1) Enter Information about patient



The screenshot shows a mobile application interface for entering patient information. At the top, there is a status bar with signal strength, time (9:41 AM), and battery level (100%). Below the status bar is a back arrow and the title 'Tell us about your patient'. A subtitle reads: 'Your individual parameters are important for monitoring your patient health with you.' The form contains four rows, each with an icon, a label, and a value or action:

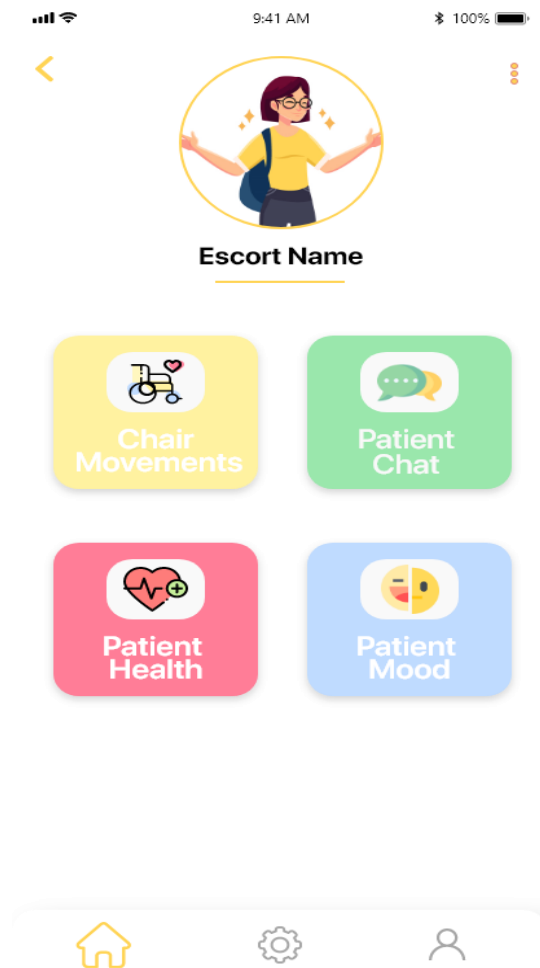
Icon	Label	Value/Action
	Patient gender	Male
	Patient weight	80 kg
	Patient age	Select
	Patient Code	Enter

At the bottom of the form is a large blue button labeled 'Next'.

(Fig.7.24) Enter Patient Information



## 2) Options for escort to help patient



(Fig.7.25) Helping Patient

## **8 Testing and Performance Evaluation :**

### **8.1 Process used for testing and evaluating your technical implementation, field deployment, etc.**

The experiment is done to five subjects, each of different ages, genders, and physical disabilities excluding the brain. The five subjects are to test each command (move forward, move left, move right and stop) for 10 times randomly. To move forward, subjects imagine an object or focus at a point in front of them and imagine that object come closer. Whereas to move left or right in EEG mode, subjects determine the direction they want to go and focus at some object on the direction they want to move. Finally, to stop the wheelchair, subjects could just blink either left or right eye.

The result of this experiment is counted from the number of true positive result from 10 tries per each movement. Tables below shows the results of the experiments without sensitivity adjustment:

**Table 8.1. Result of experiment without sensitivity adjustment**

Subject	Actions				Overall
	Forward	Stop	Left	Right	
1	9	8	8	7	
2	8	7	9	8	
3	9	10	7	8	

4	7	9	7	9	
5	10	9	8	9	
Average	8.6	8.6	7.8	8.2	

The following table shows the experiment result with the same five subjects and the same random command procedure as before with the sensitivity adjusted:

**Table 8.2. Result of experiment without sensitivity adjustment**

Subject	Actions				Overall
	Forward	Stop	Left	Right	
1	9	8	8	8	
2	9	8	9	8	
3	9	10	8	9	
4	8	9	7	8	

5	10	9	9	9	
Average	9	8.8	8.2	8.4	8.6

## 8.2 Analysis of Results and efforts to enhance reliability and performance (as applicable)

Product that will completely change the lives of its stockholders, convenient wheelchairs require patients to be escorted by someone to move, ours allow patients to move on their own, not only that, but he can also communicate with the rest of the world by writing and making emergency calls and express their opinion & feelings like healthy people. His escorts can remotely monitor his health and mood conditions, as well as move the chair. We expect the product to be purchased by 70% of targeted patients. In addition to providing material revenue to many people, including chair manufacturers, engine sellers, programmers, etc. And we will seek to improve performance by using sensor with 5 channels instead of sensor with only 3 channels.

## 9 Usability and societal impact of the product (including legal/ethical considerations) :

Electric wheelchairs have been considered as one of important mobility aids for the elderly as well as the physically impaired patients. Clinicians pointed out that approximately 50% of patients including paralyzed patients cannot be able to control an electric wheelchair by conventional methods. Especially, people getting MND can only use the eyes and brain to exercise their willpower. In the context,

EEG-controlled wheelchairs are a mobility aid especially suitable for the paralyzed patients that are unable to operate the electric wheelchair completely. When the patients have suffered from MND, their muscle does be gradually wasting and weakness and then their body is frozen. The main type of MND is named Amyotrophic Lateral Sclerosis (ALS). Because of the famous American baseball star Lou Gehrig died of this disease, it is also called Lou Gehrig's disease. Their words, swallowing and respiration are dysfunctional until respiratory failure and death. The disease can infringe upon anyone, but more common in the age of 40-70 years old. The development of the disease is rapid and ruthless. Generally, the average life expectancy of survival is between 2-5 years after onset. Because sensory nerves have not been violated, it does not affect the patient's intelligence, memory or feeling. There are numerous interfaces between human and machines to utilize input devices such as mouse, keyboards or joysticks. Recently, a number of biological signals have been utilized as hands-free interfaces, names brain-computer interface (BCI) to machines like electromyogram (EMG) and electroencephalogram (EEG). The paralyzed patients cannot operate objects or communicate their needs, even though their mental capabilities are integral. Therefore, the precious reason promoted us to design a BCI system and a drive circuit by using the paralytic patients' brain waves to control electric wheelchairs and to help them to move freely in their daily life. There is a rising tendency in the growing recognition requirements of BCI systems. The existing BCI systems focus on developing new superior control and communication technologies for those with strict neuromuscular disorders.

## **10. Conclusions and Recommendations for Future Work**

### **10.1 Ideas for new work that can enhance and/or complement your implementation and complete your vision :**

**1)** Our future plan is to make communication easier for the patio like full conversion with his escort with full keyboard.

**2)** Add entertainment to the user like Games that depend on the movement of the eye blink, Play Music ad play videos on YouTube.

**3)** Reduce the cost of the product by manufacturing wheelchair and sensor.