Switchack


SWITCHACK
Monitor your usage .. Lower your bills
MacBook
SWITCHACK
Monitor your usage .. Lower your bills

Team Members

Under Supervision of:
Dr. Shahira Habshy
Prof. Sameh Salem

1. **Radwa Nabil**
   (Hardware Design and implementation)
2. **Nada Ahmed**
   (Android Development)
3. **Karim Hashim**
   (Hardware Design and implementation)
4. **Mohamed El-Sayed**
   (Machine Learning & Hardware Design and implementation)
5. **Jannat Saeed**
   (Android Development)
6. **Hesham Gamal**
   (IOS Development)
7. **Alaa Hussien**
   (Android Development)
8. **Mohammed Osama**
   (Communication between software and hardware)

Competitions:

1. Third place in Maker Hackathon Cairo
2. One free slot in Cairo Maker Faire
3. Participants at MIE competition

Outlines

1. Introduction
2. Proposed Solution
   - A. Project overview
   - B. System features
   - C. System Architecture
3. System modules
   - A. Hardware module
   - B. Database server
   - C. Software module
     - Android Application
     - IOS Application
   - D. Machine learning
4. Marketing
5. Time plan

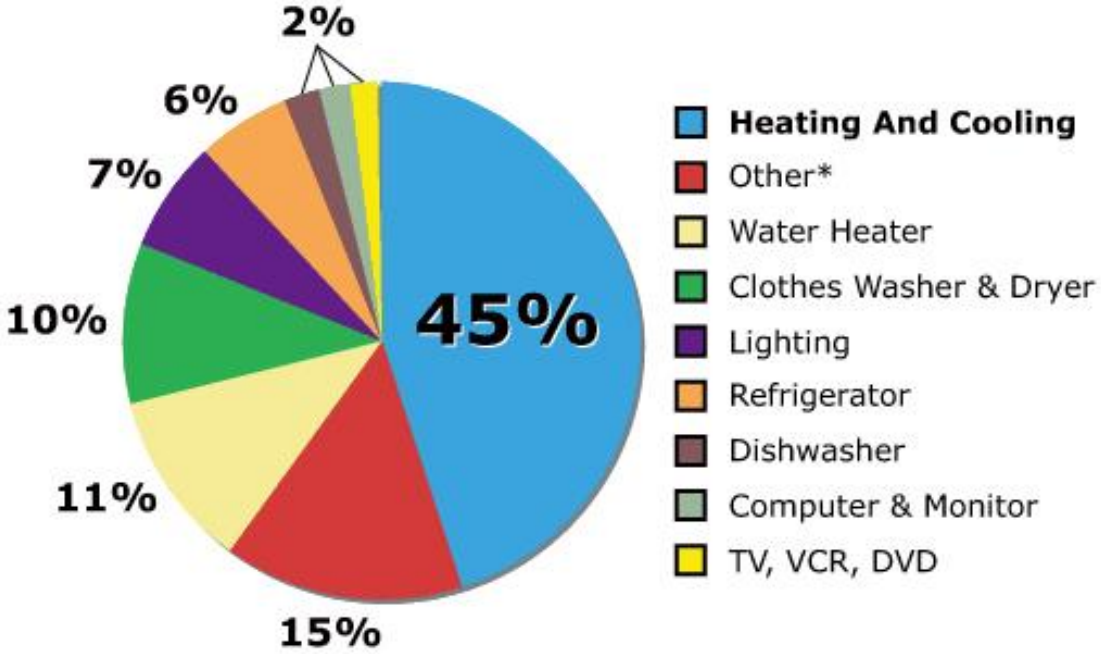
Problem definition

01- Increasing the Electricity prices in Egypt

Where prices increased by an average of 33% for all segments in july 2017.

02- Wasting Electricity

Reducing energy use in your home saves you money, increases our energy security and reduces the pollution that is emitted from non-renewable sources of energy.

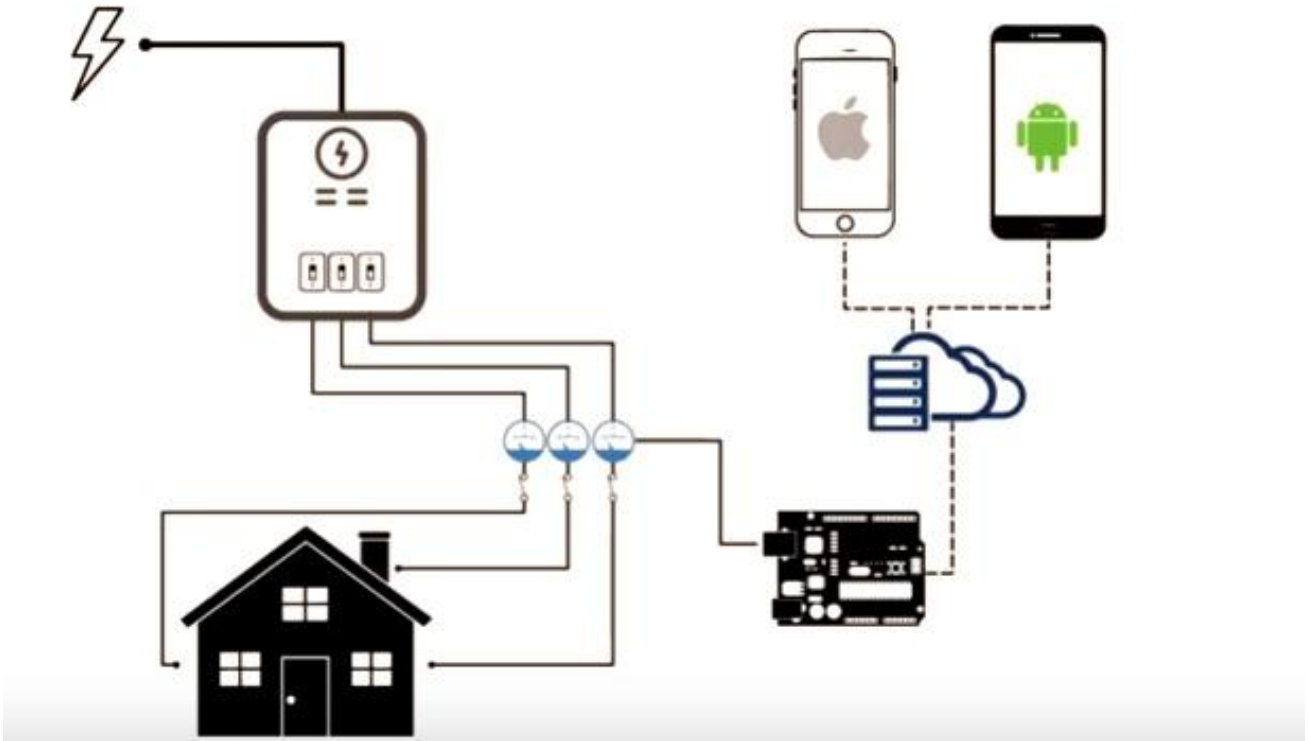


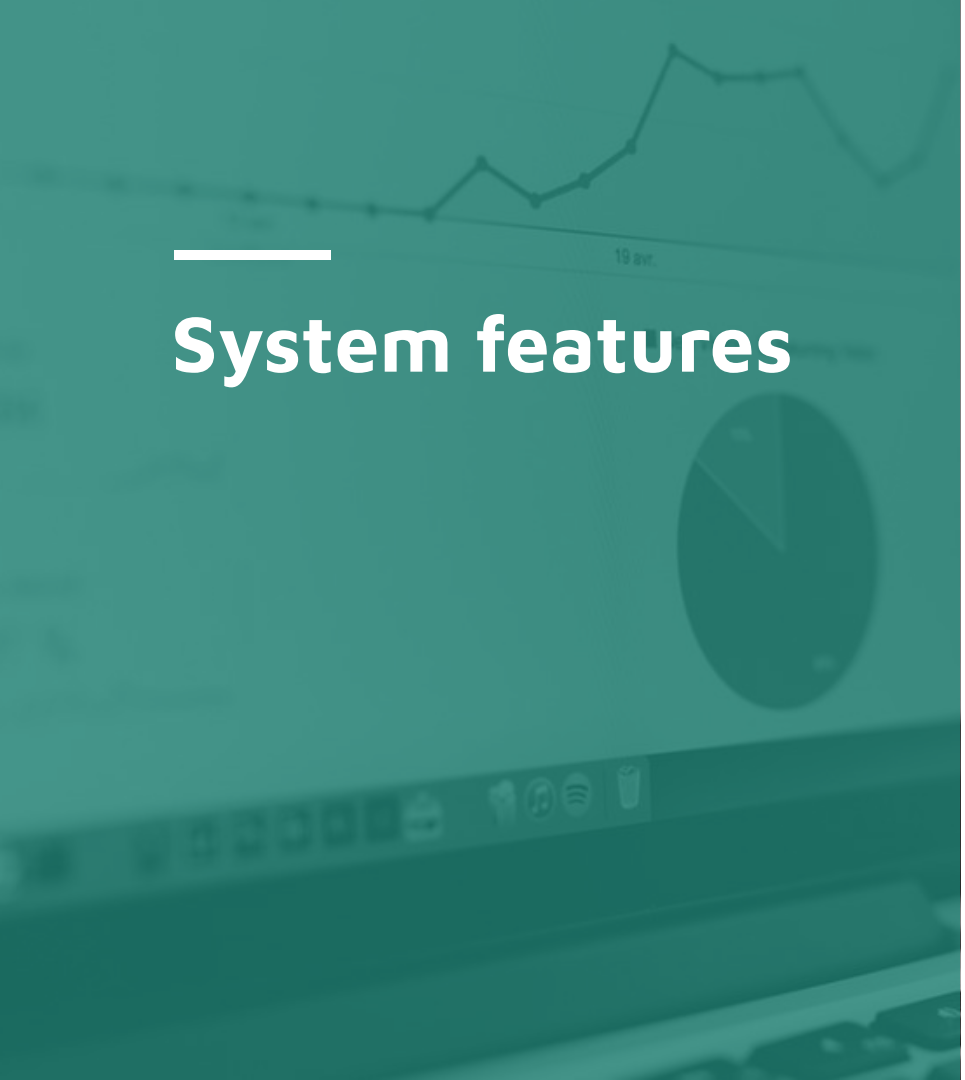
Outlines

1. Introduction
2. Proposed Solution
   - A. Project overview
   - B. System features
   - C. System Architecture
3. System modules
   - A. Hardware module
   - B. Database server
   - C. Software module
     - Android Application
     - IOS Application
   - D. Machine learning
4. Marketing
5. Time plan

Project Overview:

We propose to solve this critical problem using a user friendly mobile application and a smart device could be installed at your home besides your electrical panel to measure your usage.





System features

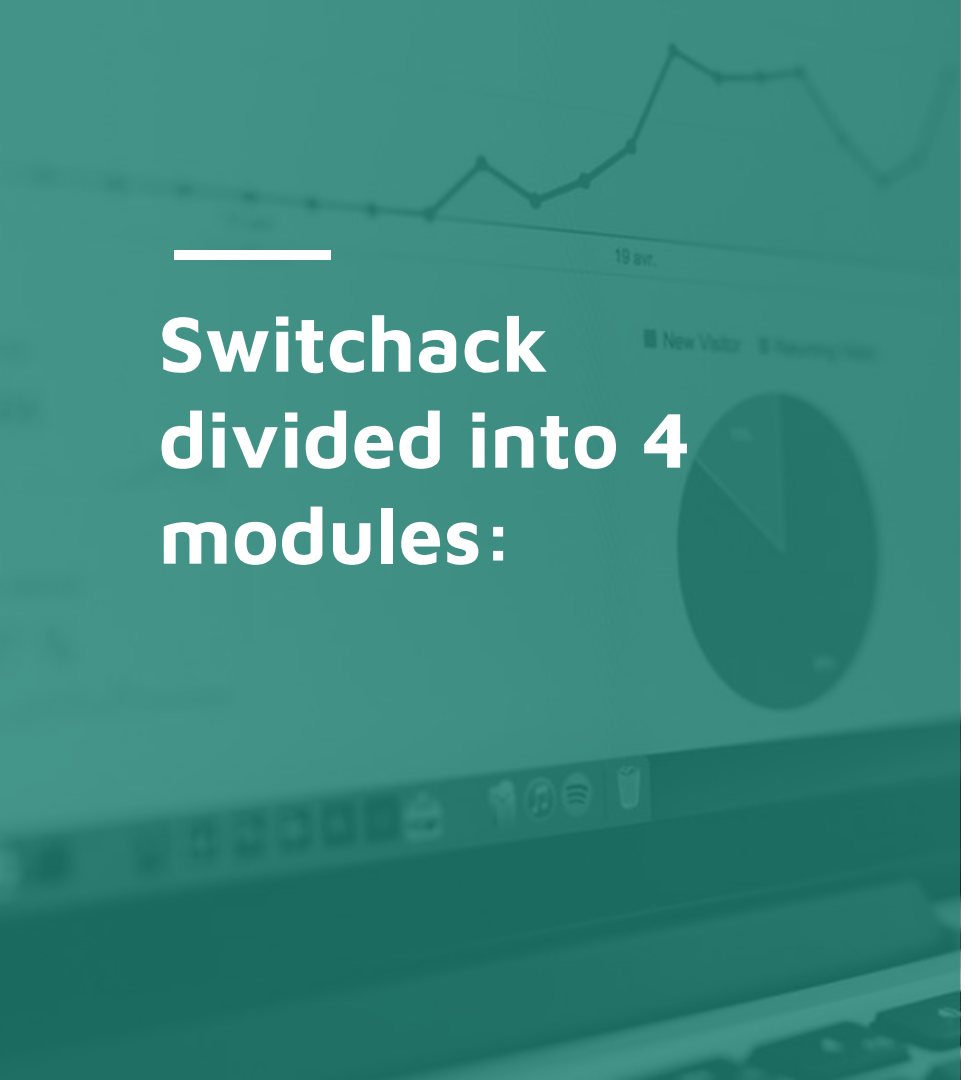
1. Measuring consumed current
2. Monitoring your usage
3. Controlling fuses & some devices
4. Estimating electricity bills
5. Recommending how to economize your usage

System Architecture:

Client- Server

Outlines

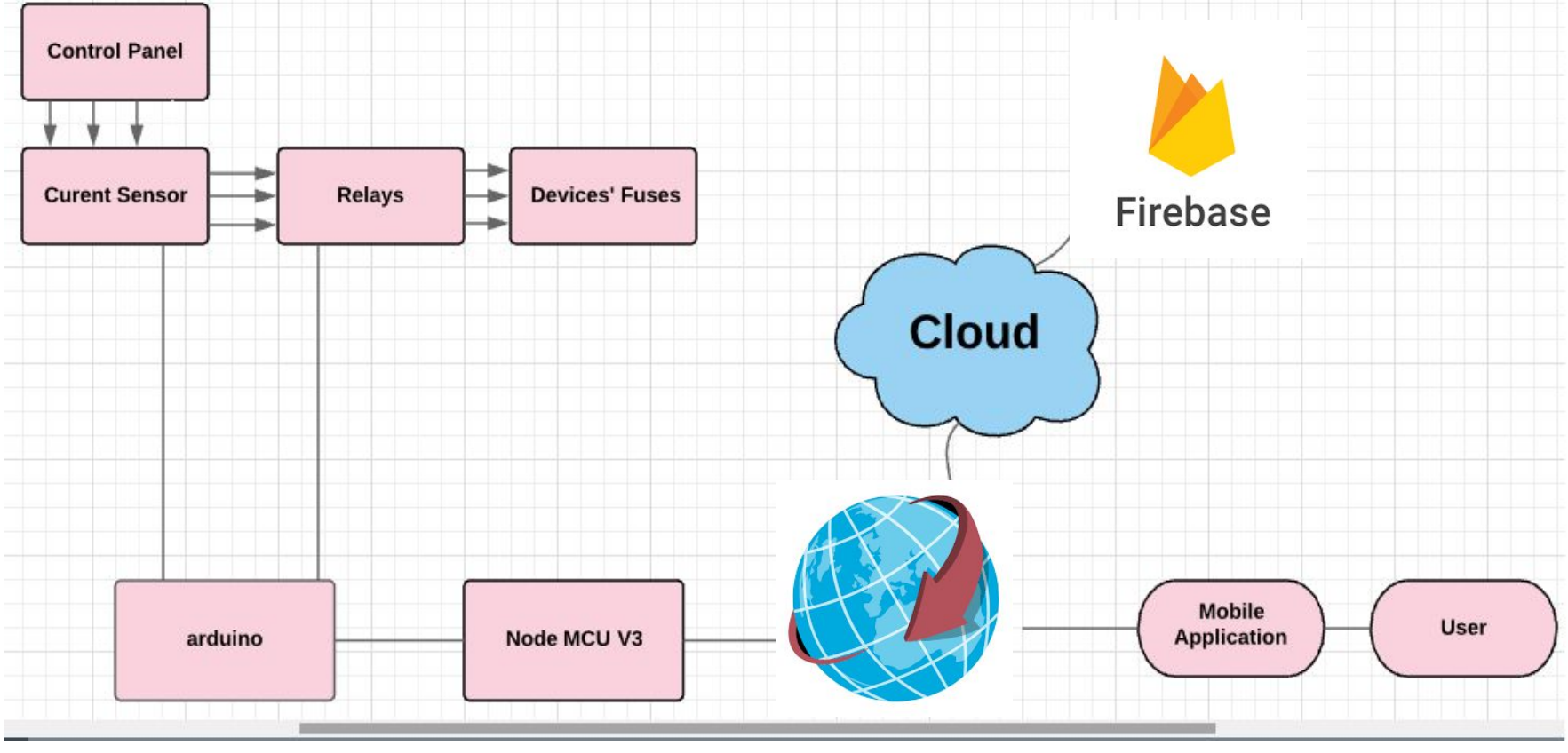
1. Introduction
2. Proposed Solution
   - A. Project overview
   - B. System features
   - C. System Architecture
3. System modules
   - A. Hardware module
   - B. Database server
   - C. Software module
     - Android Application
     - IOS Application
   - D. Machine learning
4. Marketing
5. Time plan



Switchack divided into 4 modules:

1. Hardware module
2. Software module (Android / IOS)
3. Database Server (Firebase)
4. Machine learning module

System Modules



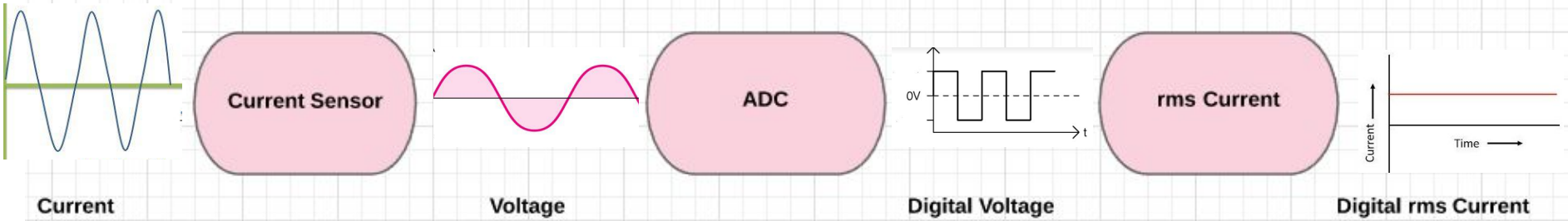
01- Hardware module

Hardware functions:

1. Measure consumed current
2. Measure Ac voltage
3. Measure power factor
4. Send readings to cloud database (Firebase)
5. Controlling Fuses(and some devices)

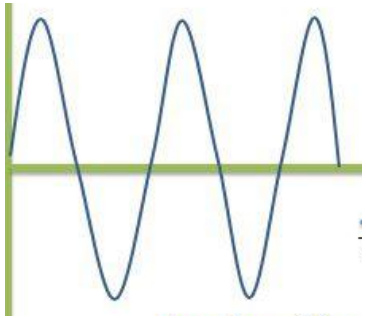
Hardware Implementation

A) Measure AC current:



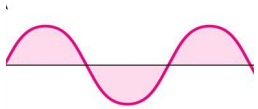
B) Measure AC Voltage:

1. Reduce the AC voltage to acceptable level less than 5 volts



High AC Voltage

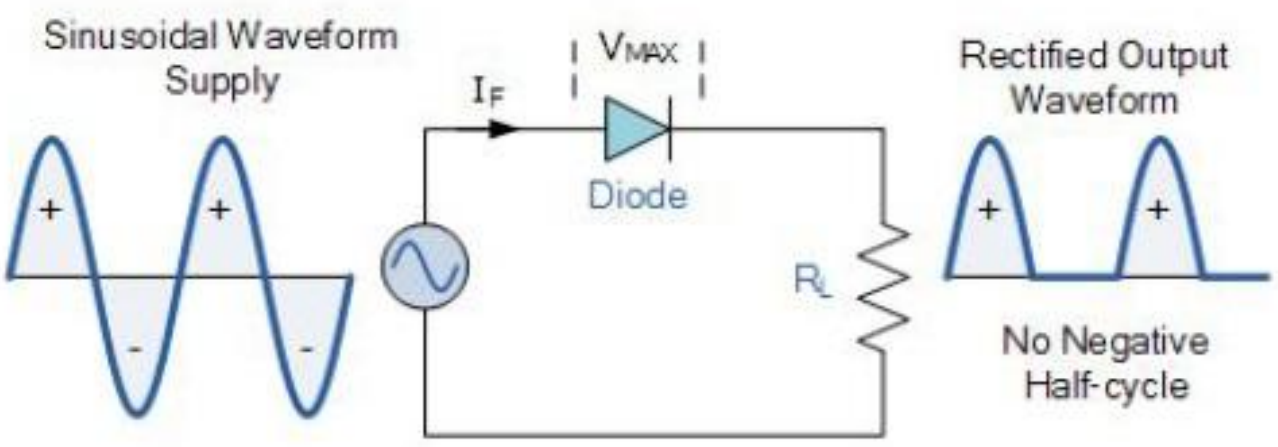
Voltage divider



Low AC voltage

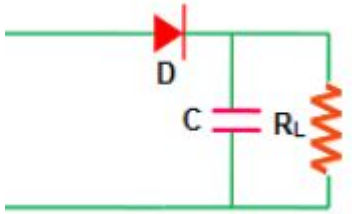
B) Measure AC Voltage:

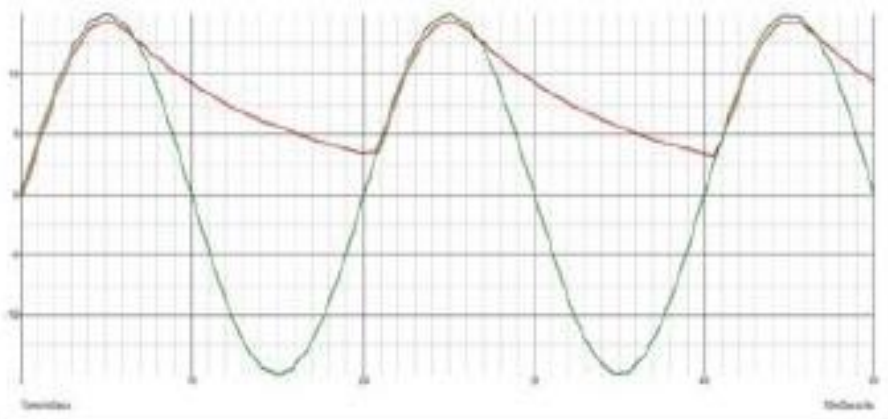
2. half wave rectifier to take only positive cycle



B) Measure AC Voltage:

3. Capacitor is connected to smooth out the ac signal because it contain large ripples.





Input and Output Waveforms

B) Measure AC Voltage:

4. Calculate voltage with Arduino Read analog value at pin A0

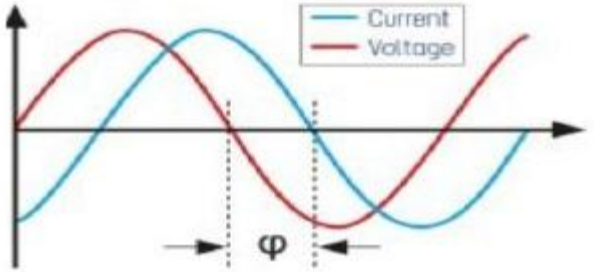
Ac Volt = analog value * 429 / 1024

C) Measure Power factor:

1. Calculate the time difference (td) between two waves and calculate period = 1/frequency.

2. Phase shift (Φ) = td *360 / period
Where Φ is the phase angle between voltage and current

3. Then calculate power factor = cos (Φ)



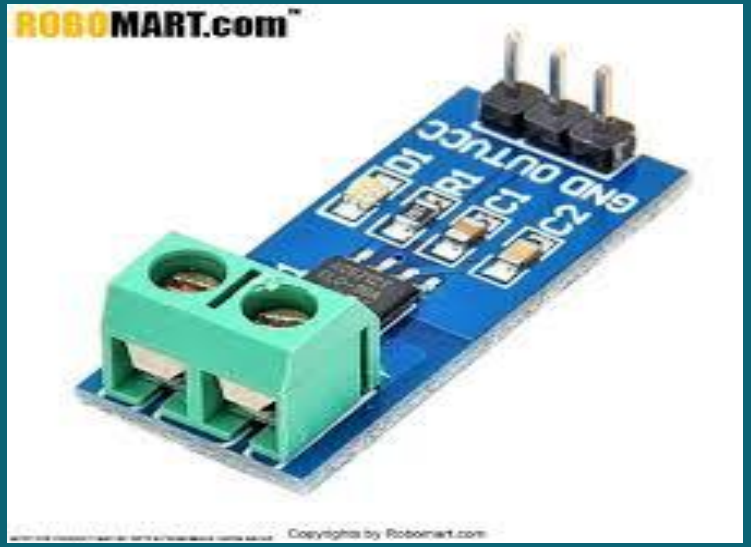
C) Measure Active Power:

Power = current * voltage * cos(Φ)

Hardware Components

Current sensor

Measure consumed current (per sec) into the wire of each fuse

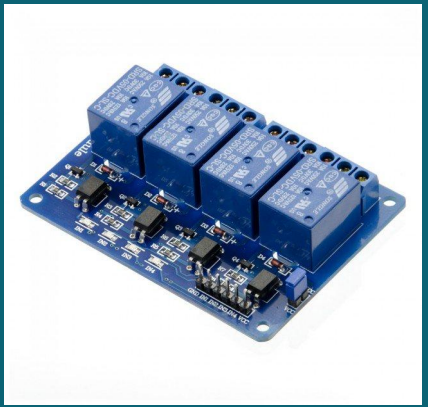


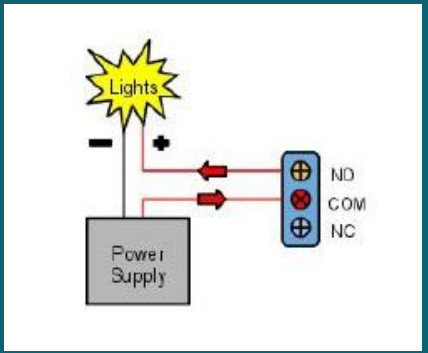
Hardware Components

Relay

Responsible for controlling ON/OFF

operation of fuses

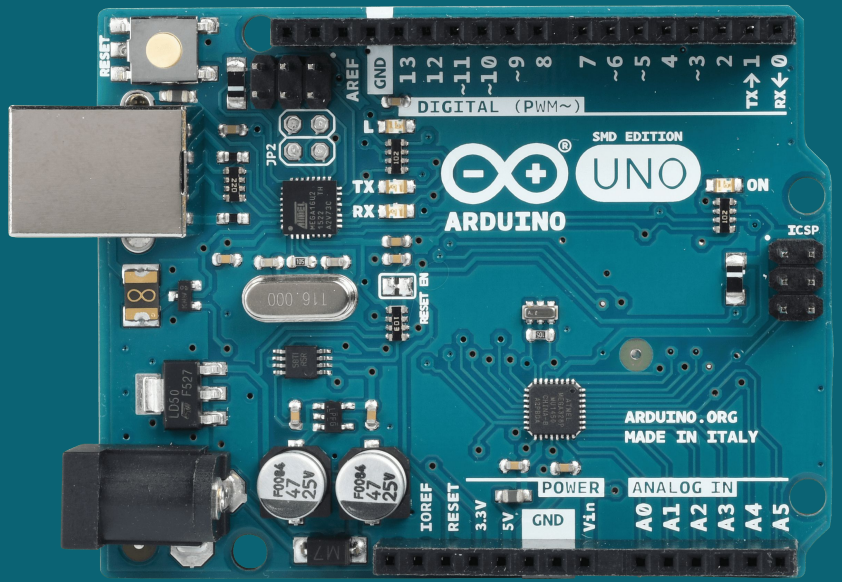




Hardware Components

ARDUINO UNO

Responsible for retrieving data from current sensors & relays and convert it from analog to digital then pass it to wifi module.



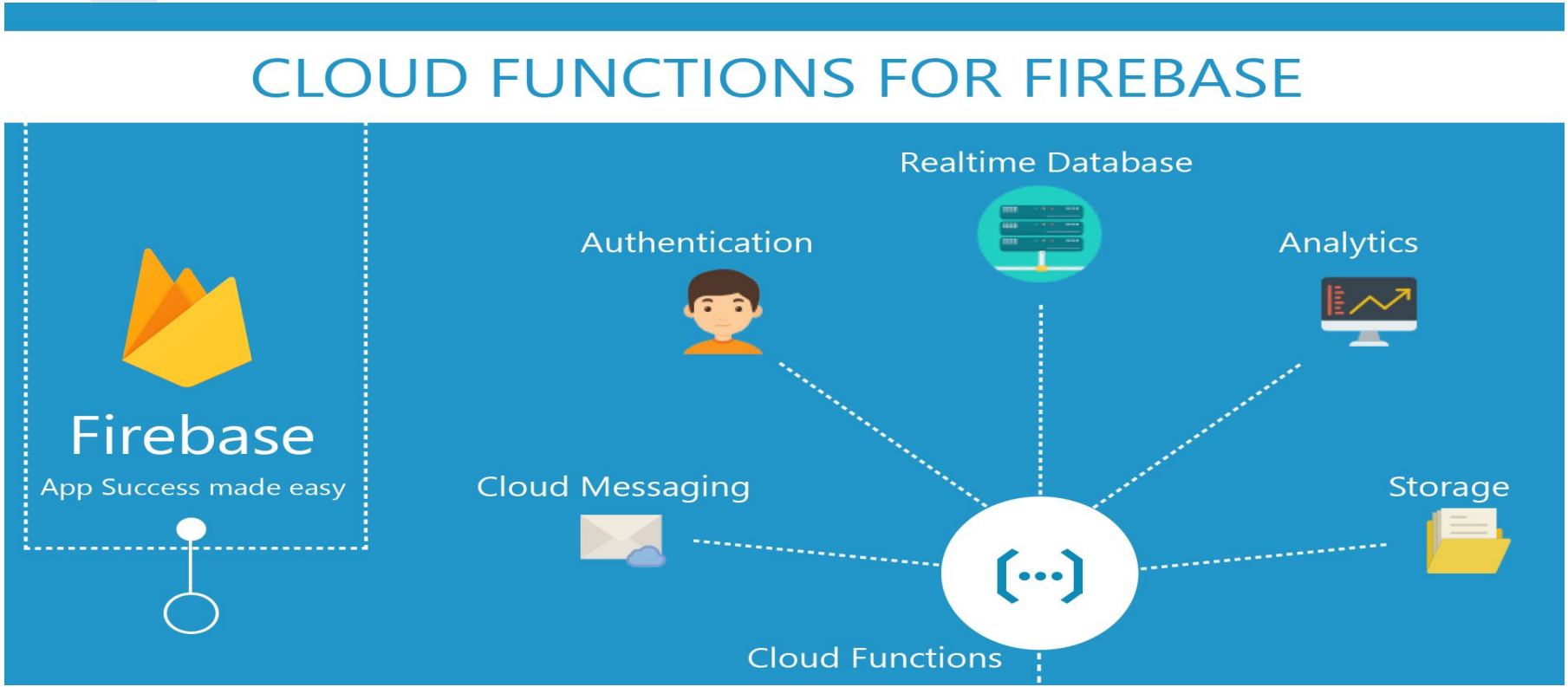
Hardware Components

NodeMCU V3

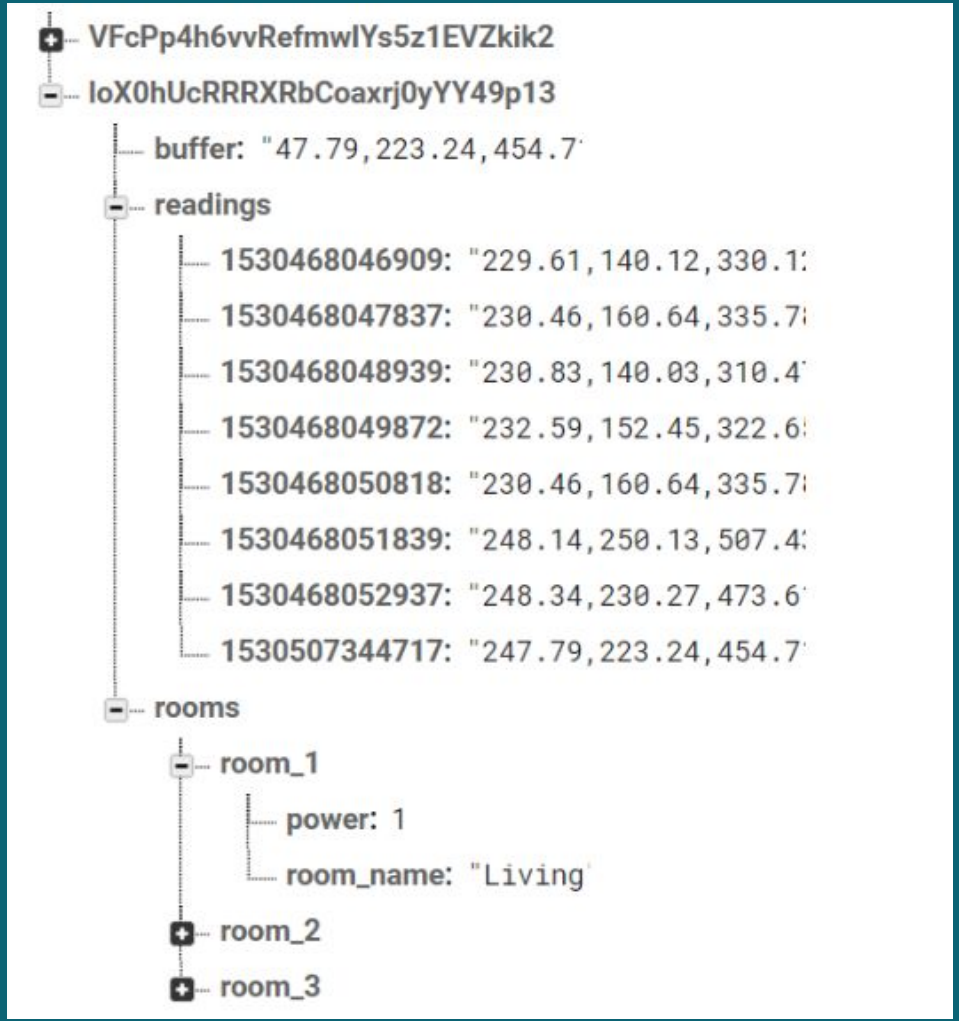
Sending collecting reading to cloud database (firebase)

02- Database

Why Firebase ?



Database Structure:



Rooms Entity:

This entity refers to the rooms in the apartment using our system, each room has two parameter, Power which indicate this fuse is ON or OFF, and room name which is editable by the user.

JSON code for rooms table:

```
"rooms" : {
  "room_1" : {
    "power" : 1,
    "room_name" : "Bedroom"
  },
  "room_2" : {
    "power" : 1
  }
```

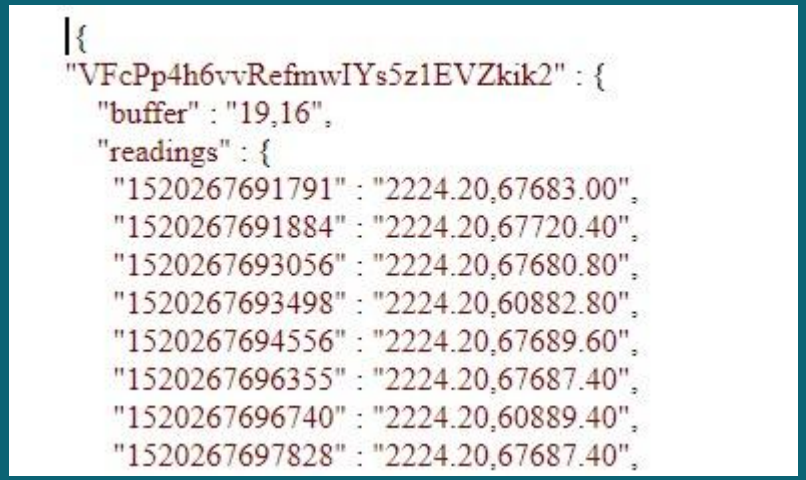

```
{
"VFcPp4h6vvRefmwIYs5z1EVZkik2" : {
  "buffer" : "19,16",
  "readings" : {
    "1520267691791" : "2224.20,67683.00",
    "1520267691884" : "2224.20,67720.40",
    "1520267693056" : "2224.20,67680.80",
    "1520267693498" : "2224.20,60882.80",
    "1520267694556" : "2224.20,67689.60",
    "1520267696355" : "2224.20,67687.40",
    "1520267696740" : "2224.20,60889.40",
    "1520267697828" : "2224.20,67687.40",
```


Readings Entity:

This entity carries all the readings and act like real time database, each reading also related to certain room.

Readings Entity:

This entity carries all the readings and act like real time database, each reading also related to certain room.



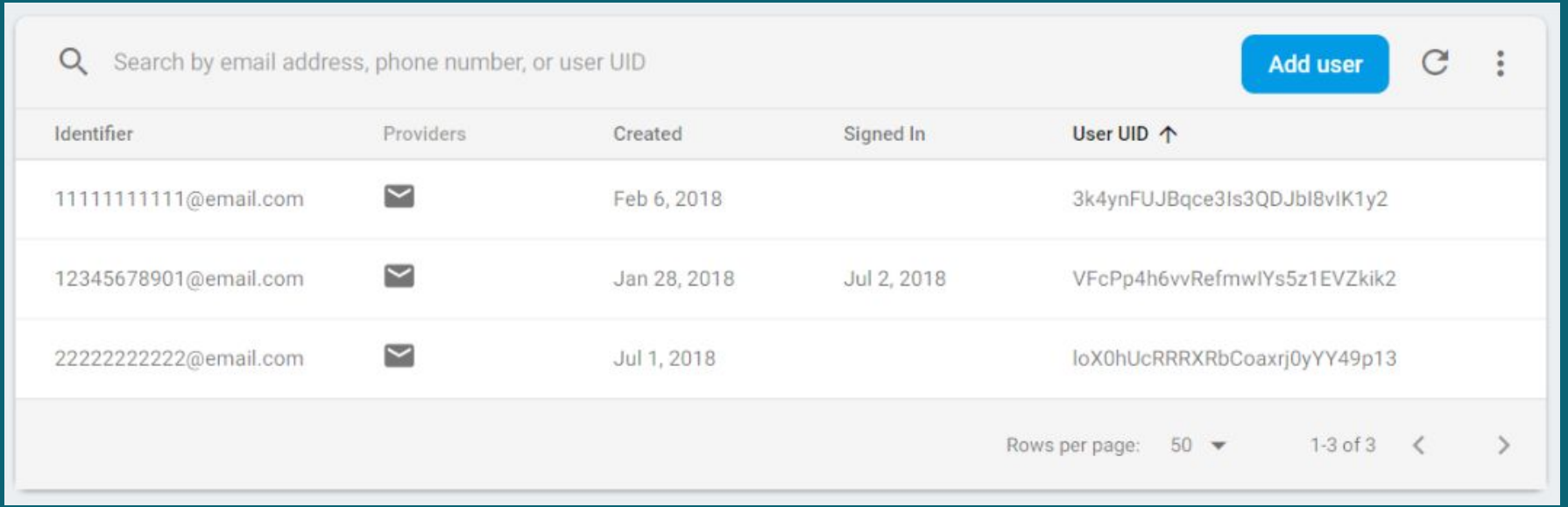

```
{
"VFcPp4h6vvRefmwIYs5z1EVZkik2" : {
  "buffer" : "19,16",
  "readings" : {
    "1520267691791" : "2224.20,67683.00",
    "1520267691884" : "2224.20,67720.40",
    "1520267693056" : "2224.20,67680.80",
    "1520267693498" : "2224.20,60882.80",
    "1520267694556" : "2224.20,67689.60",
    "1520267696355" : "2224.20,67687.40",
    "1520267696740" : "2224.20,60889.40",
    "1520267697828" : "2224.20,67687.40",
```


Why do we use “Buffer”?

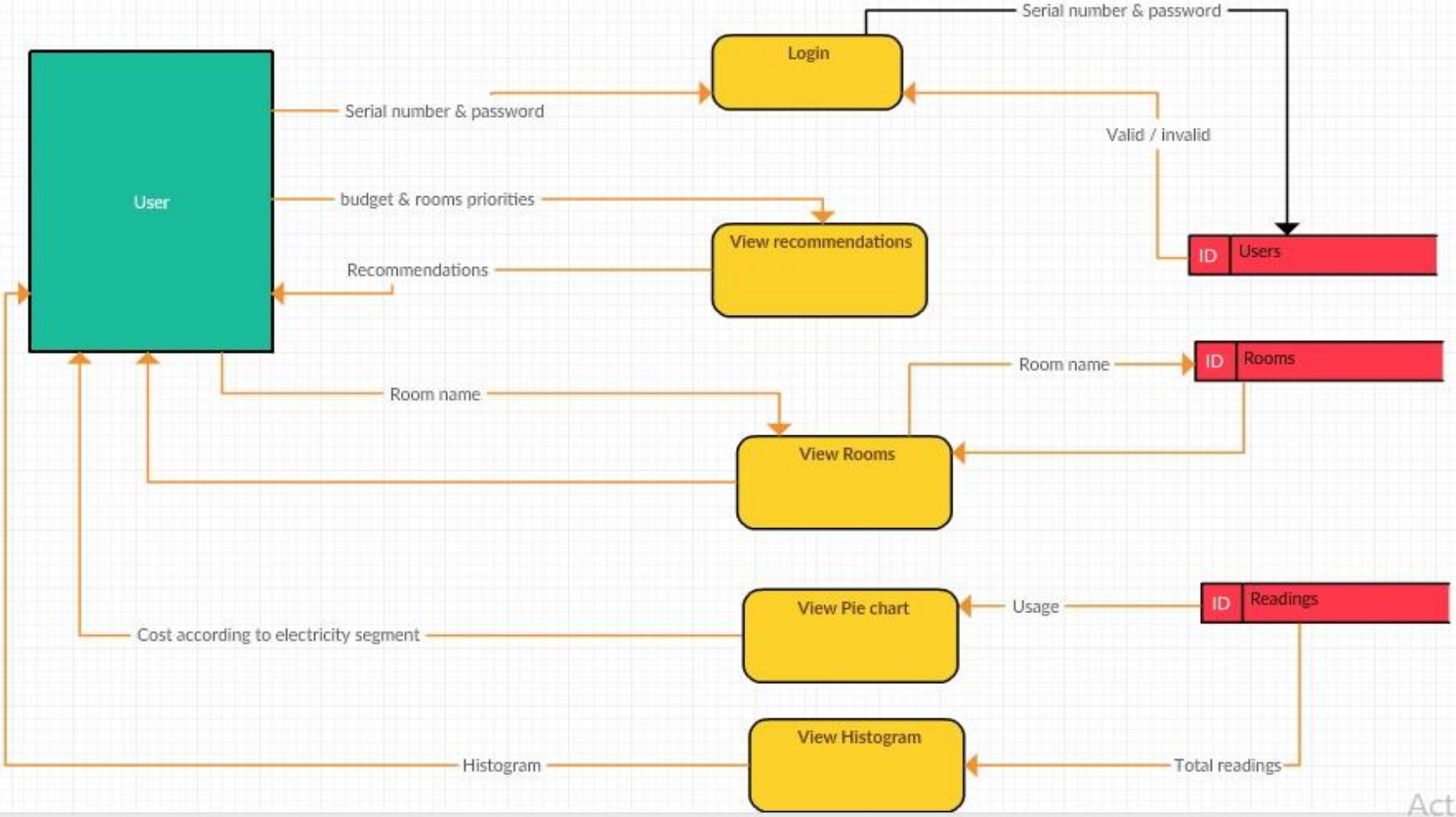
Since Arduino is unaware of time, so it just send the reading instantly to firebase, we deal with this problem by using buffer which receives the readings from arduino and adds timestamp to it and then passes it to readings entity.

Note: This functionality is implemented using firebase cloud functions. By continuously checking for edits on buffer and then creating a key value pair in readings whenever the buffer value changes

Database authentication file:



Data flow Diagram:

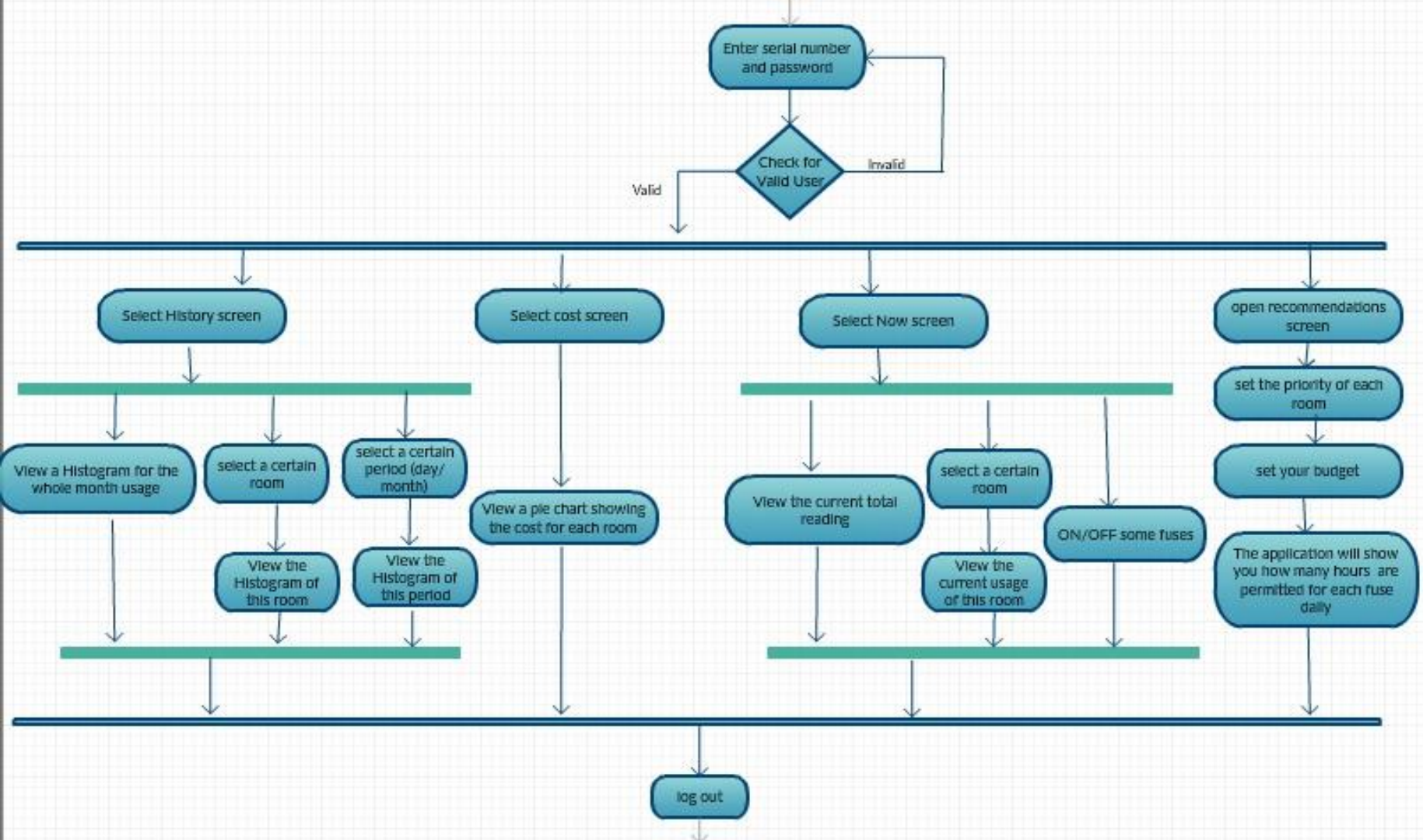


03- Software module

Software functions:

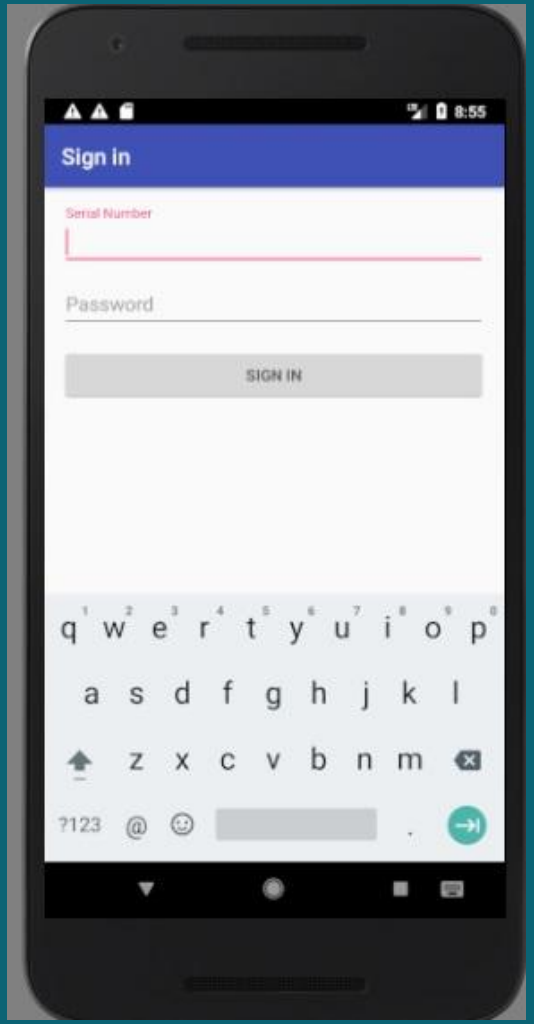
1. Monitoring usage (Using Histogram)
2. Estimating costs (Using Pie chart)
3. Controlling some fuses/devices (ON/OFF)
4. Recommending how to reduce your usage

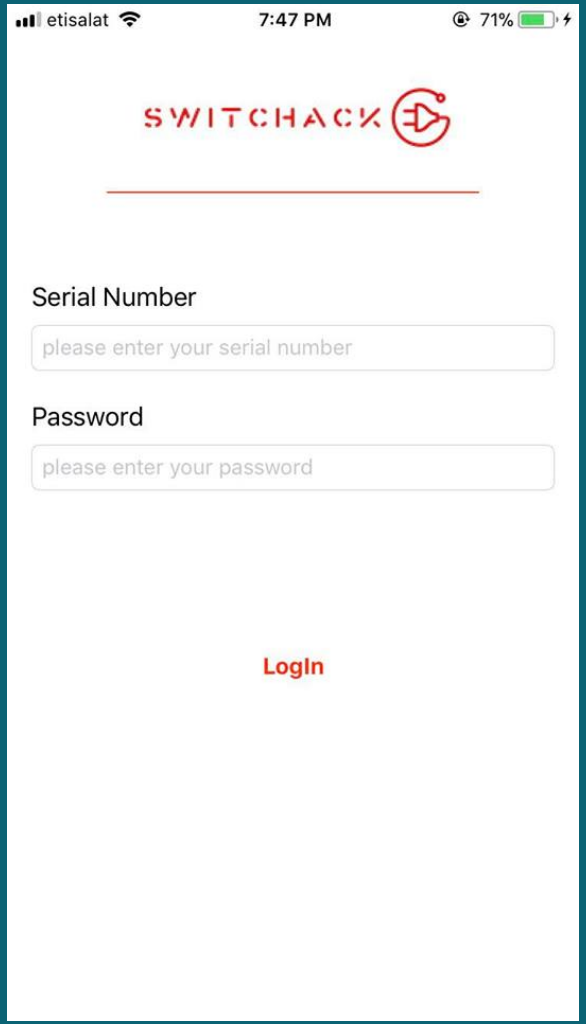
Software Activity Diagram:


Enter serial number and password
Check for Valid User
Invalid
Valid
Select History screen
Select cost screen
Select Now screen
open recommendations screen
View a Histogram for the whole month usage
select a certain room
select a certain period (day/ month)
View the Histogram of this room
View the Histogram of this period
View a pie chart showing the cost for each room
View the current total reading
select a certain room
ON/OFF some fuses
View the current usage of this room
set the priority of each room
set your budget
The application will show you how many hours are permitted for each fuse daily
log out

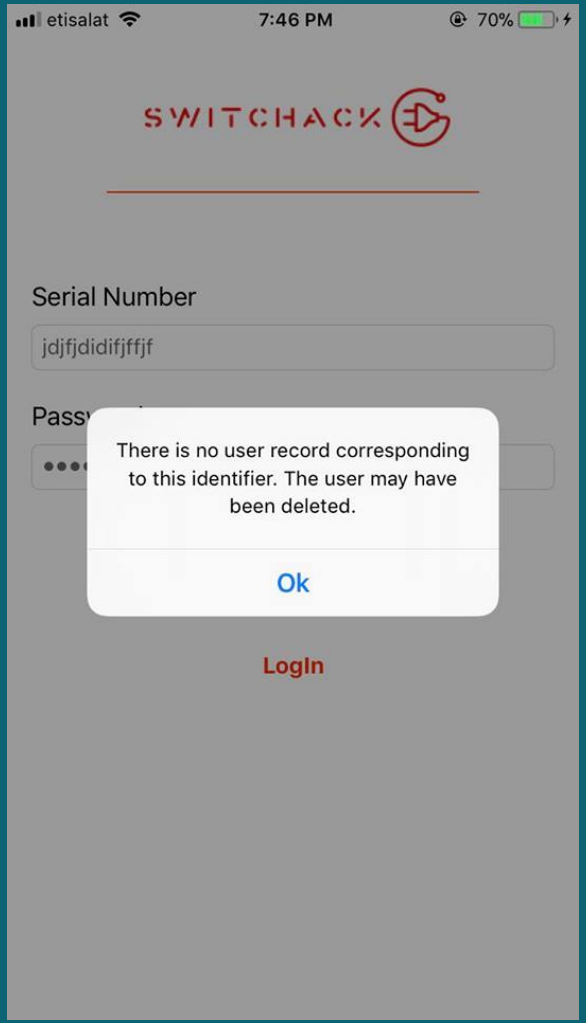
Mobile application Screens:

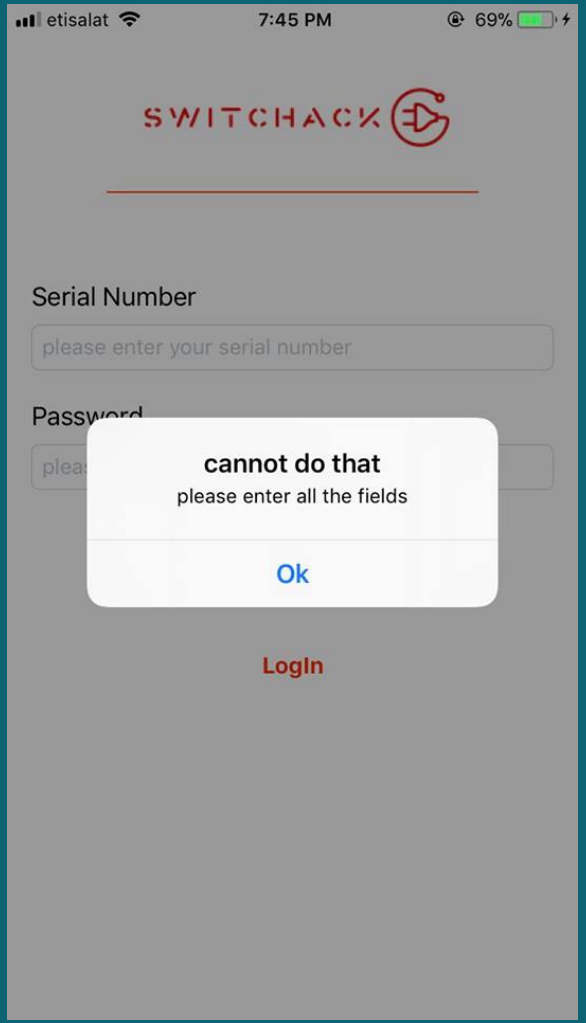
Login Screen



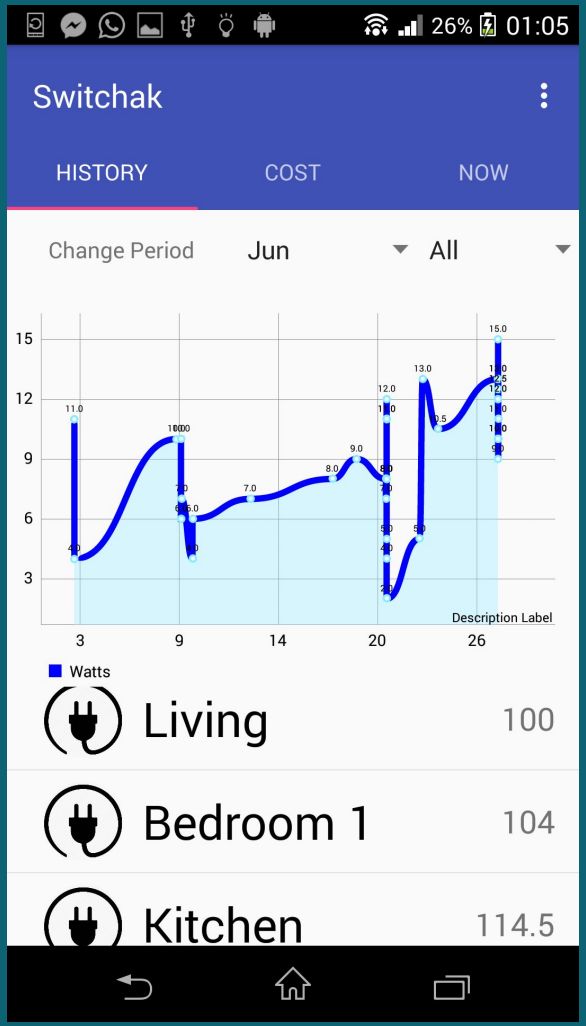


Login Screen





History Screen

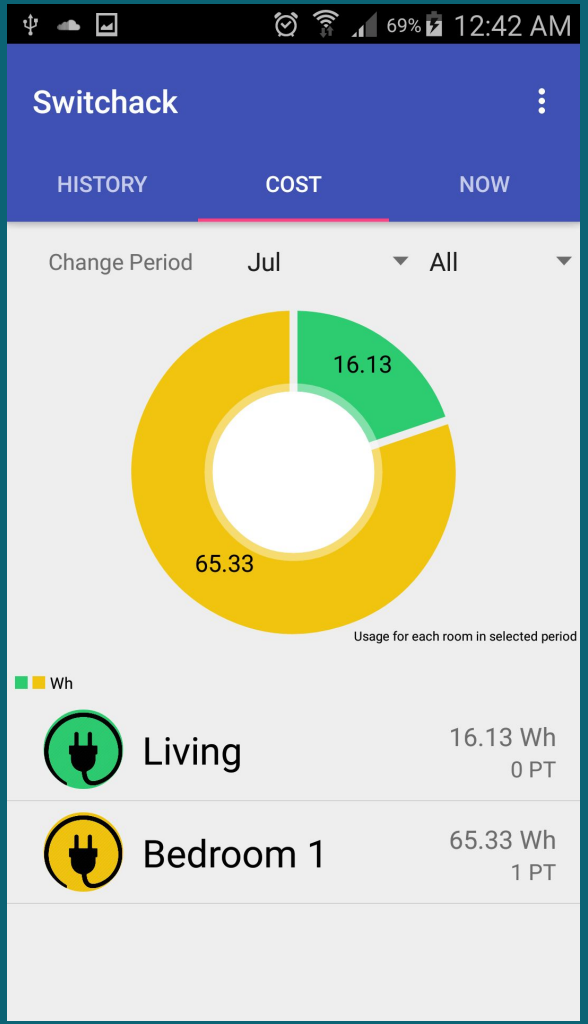


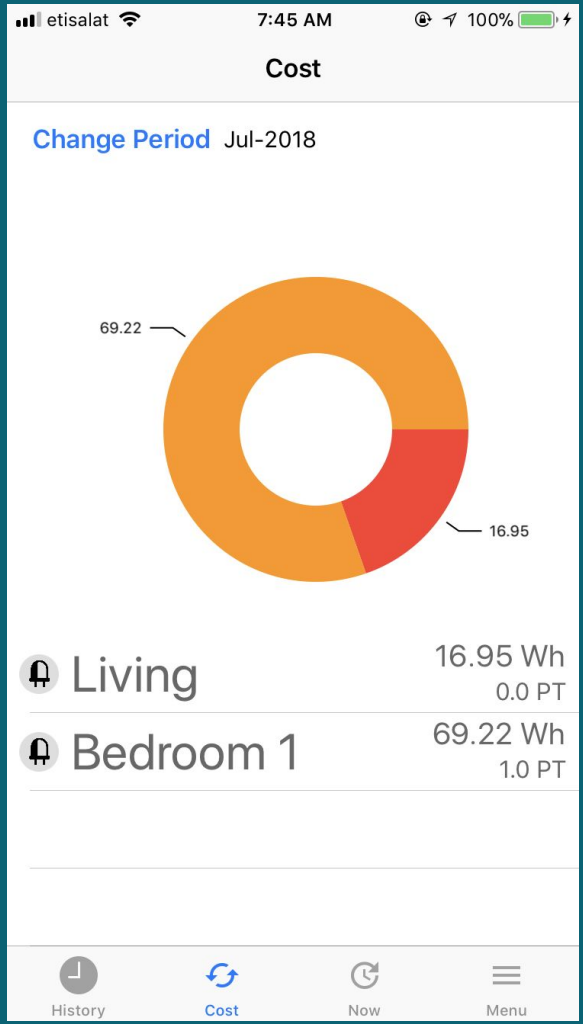
History Screen

In History screen we retrieve readings & timestamp of each reading and build a Histogram so the user can easily monitor his usage during the month or during any period he determined.

```
Fragment changePeriodFragment = new ChangePeriodFragment();
FragmentTransaction transaction = getChildFragmentManager().beginTransaction();
transaction.add(R.id.fragment_change_period, changePeriodFragment);
transaction.commit();

XAxis xAxis = chart.getXAxis();
xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);

xAxis.setValueFormatter((value, axis) -> {
        Calendar c = new GregorianCalendar();
        c.setTime(new Date((long) value));

        String hour = String.valueOf(c.get(Calendar.HOUR));
        int amPm = c.get(Calendar.AM_PM);
        String day = String.valueOf(c.get(Calendar.DAY_OF_MONTH));
        if (chart.getHighestVisibleX() - chart.getLowestVisibleX() < 86400000 *
            //milliseconds in a day = 86400000
            return day + " " + hour + (amPm > 0 ? "PM" : "AM");
        else return day;
});

YAxis yAxis = chart.getAxisLeft();

YAxis rightAxis = chart.getAxisRight();
rightAxis.setEnabled(false);

  yAxis.setAxisMinimum(0);

lineDataSet = House.getInstance().getDataSet();
lineDataSet.setMode(LineDataSet.Mode.HORIZONTAL_BEZIER);
lineDataSet.setColor(Color.BLUE);
```

Cost Screen


69% 12:42 AM
Switchack
HISTORY
COST
NOW
Change Period
Jul
All
16.13
65.33
Usage for each room in selected period
Wh
Living
16.13 Wh
0 PT
Bedroom 1
65.33 Wh
1 PT


etisalat
7:45 AM
100%
Cost
Change Period
Jul-2018
69.22
16.95
Living
16.95 Wh
0.0 PT
Bedroom 1
69.22 Wh
1.0 PT
History
Cost
Now
Menu

Cost Screen

Using cost screen we help user to know the calculated cost for each room according to his electricity segment, and use pie chart to be more friendly to user.

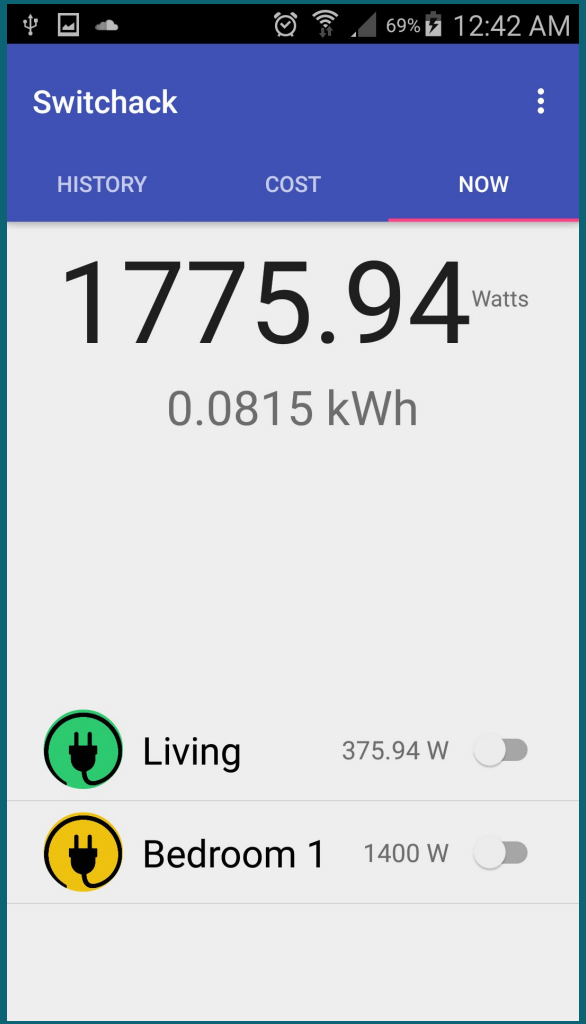
```
public float getCostFromUsage(float usage) {
    float value = House.getInstance().getThisMonthReading() / (3600 * 1000);
    float cost = 0;
    usage = usage / (1000 * 3600);

    if (value >= 0 && value <= 50)
        cost = usage * 0.13f;
    else if (value >= 51 && value <= 100)
        cost = usage * 0.22f;
    else if (value > 100 && value <= 200)
        cost = usage * 0.22f;
    else if (value > 200 && value <= 350)
        cost = usage * 0.45f;
    else if (value > 350 && value <= 650)
        cost = usage * 0.55f;
    else if (value > 650 && value <= 1000)
        cost = usage * 0.95f;
    else if (value > 1000)
        cost = usage * 1.35f;
    return cost;
}
```

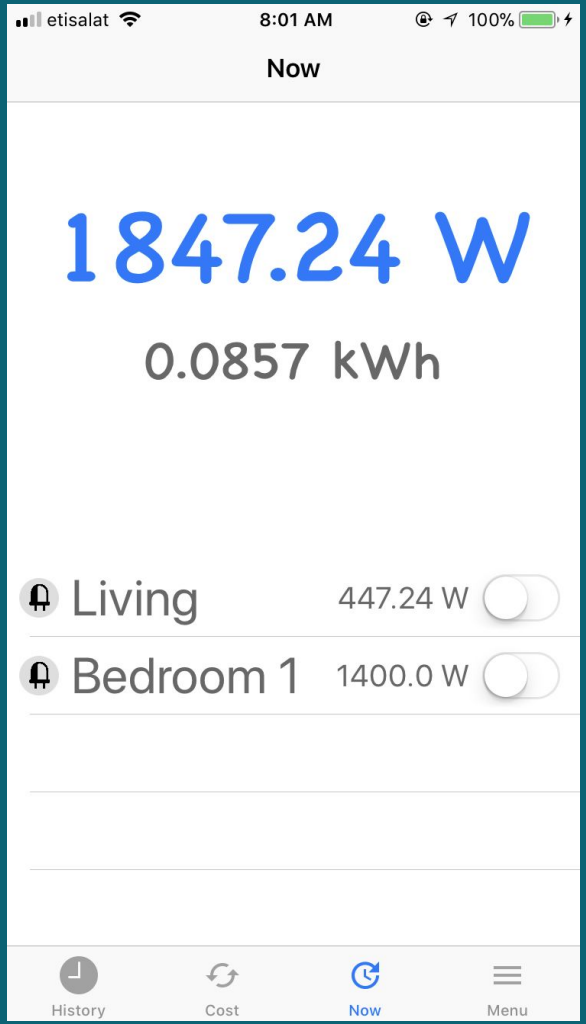
Cost Screen

User can also change and select certain period of time.

```
//If user changed period
if (arg != null && (int) arg == FirebaseUtils.PERIOD_CHANGED) {
    beginningTime = FirebaseUtils.getInstance().getBeginningTime();
    endTime = FirebaseUtils.getInstance().getEndTime();
    for (int i = 0; i < House.getInstance().getRooms().size(); i++) {
        float roomIReading = 0;
        for (int j = 0; j < House.getInstance().getRooms().get(i).getReadings().size(); j++)
            float timeOfJReading = House.getInstance().getEntries().get(j).getX();
            if (timeOfJReading >= beginningTime && timeOfJReading < endTime)
                roomIReading = roomIReading + House.getInstance().getRooms().get(i).getReadi
        }
        pieEntries.set(i, new PieEntry(getCostFromUsage(roomIReading)));
        pieEntries.set(i, new PieEntry(roomIReading));
        FirebaseUtils.getInstance().getRooms().get(i).setSelectedPeriodReading(roomIReading)
    }
    pieData.notifyDataChanged();
    pieChart.notifyDataSetChanged();
    pieChart.animateXY( durationMillisX: 1000,  durationMillisY: 1000);
    FirebaseUtils.getInstance().update();
}
mAdapter.notifyDataSetChanged();
```

Now Screen


69% 12:42 AM
Switchack
HISTORY
COST
NOW
1775.94 Watts
0.0815 kWh
Living 375.94 W
Bedroom 1 1400 W


etisalat
8:01 AM
100%
Now
1847.24 W
0.0857 kWh
Living 447.24 W
Bedroom 1 1400.0 W
History
Cost
Now
Menu

Now Screen

Using now screen we help the user to know his current usage at the moment, so he if he finds that a certain room/ device is consume more than needed he can easily cut off electricity in this fuse.

```swift
// MARK: UITableViewDelegate, UITableViewDataSource
extension NowVC: UITableViewDelegate, UITableViewDataSource{
    func numberOfSections(in tableView: UITableView) -> Int {
        return 1
    }

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return nowRooms.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath) as! NowCell

        let room = nowRooms[indexPath.row]
        cell.roomNameLabel.text = room.room_name
        cell.switchButton.setOn(room.power, animated: true)
        cell.switchButton.tag = indexPath.row
        cell.switchButton.addTarget(self, action: #selector(onTapSwitchButton(_:)), for: .touchUpInside)

        return cell
    }
}
```

Android Application:

Programming language used : Java

User Interface language used : XML

Animation & Graphics: MP android chart

Minimum SDK: API 14 - Android 4.0 (IcecreamSandwich)

IOS Application:

Programming language used : SWIFT 4

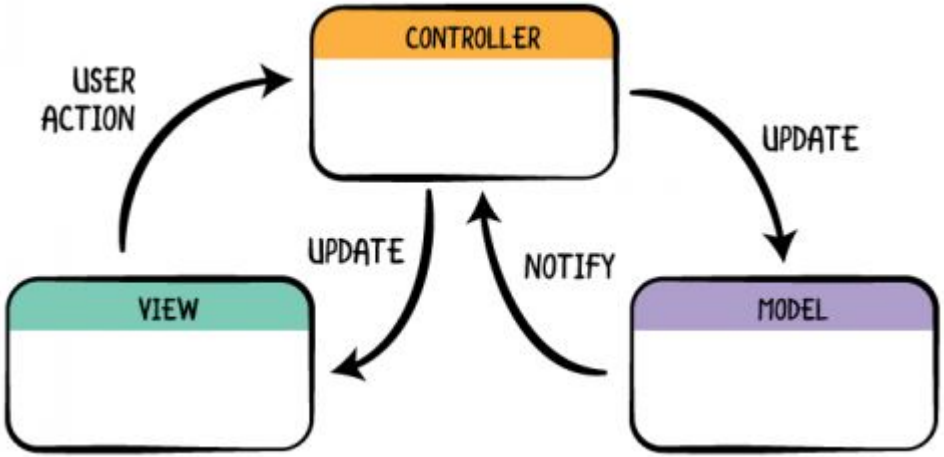
Cores used: 1. Core animation
2. Core graphics
3. Core data

Integration between IOS & Firebase: Cocoapods

Mobile Application Architecture:

Architecture:

Model–view–controller (MVC) is an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. The MVC design pattern decouples these major components allowing for efficient code reuse .



Model-view-controller (MVC)

1. Here is our model including the Singleton variable:

```swift
//
//  RoomsModel.swift
//  GraduationProject
//
//  Created by hesham ghalaab on 6/27/18.
//  Copyright © 2018 hesham ghalaab. All rights reserved.
//

import Foundation

class RoomModel {
    static var rooms = [RoomModel]()

    var id: String
    var power: Bool
    var room_name: String
    var totalReadingsOfRoom: Double
    var totalCostOfRoom: Double

```

Model-view-controller (MVC)

2. We get the data and then make our array of objects from the model.

```
94
95  func getRooms(){
96      // networking Layer is here
97      let ref = Database.database().reference()
98      ref.child(Defaults.userId).child("rooms").observe(.value, with: { (snapShot) in
99          if !snapShot.exists(){
100             self.alertUser(withTitle: "Oubs!", message: "error occured, please try again later")
101             return
102         }
103
104         let values = snapShot.value as! NSDictionary
105         self.nowRooms = []
106         for value in values{
107             let id = value.key as? String ?? ""
108             let _roomValue = value.value as! NSDictionary
109             let power = _roomValue["power"] as? Bool ?? false
110             let room_name = _roomValue["room_name"] as? String ?? ""
111             print("room_name: \(room_name), power: \(power), roomId: \(id)")
112
113             // creating our array of object
114             let room = RoomModel(id: id, room_name: room_name, power: power, totalReadingsOfRoom: 0, totalCostOfRoom: 0)
115             self.nowRooms.append(room)
116         }
117
118         self.tableView.reloadData() // updating the view
119     }) { (error) in
120         Helper.hideLoading(self.view)
121         self.alertUser(withTitle: "Oubs!", message: error.localizedDescription)
122     }
123 }
124
```

Model-view-controller (MVC)

3. Now we are ready to update our view with data

Integration between ios and other frameworks

This is done by creating a pod file that have all the frameworks we need and then handle it by cocoapods.

Pods 〉 Podfile 〉 No Selection

```
# Uncomment the next line to define a global platform for your project
# platform :ios, '9.0'

target 'GraduationProject' do
  # Comment the next line if you're not using Swift and don't want to use dynamic frameworks
  use_frameworks!

  # Pods for GraduationProject
  pod 'IQKeyboardManagerSwift'
  pod 'Firebase/Core'
  pod 'Firebase/Database'
  pod 'Firebase/Auth'
  pod 'Firebase/Storage'
  pod 'Firebase/Messaging'

  pod 'SwiftCharts', '~> 0.6.1'
  pod 'PieCharts'

end
```

Human Interface Guidelines

4. we made our iOS application according to Human interface guideline

iOS Design Themes

As an app designer, you have the opportunity to deliver an extraordinary product that rises to the top of the App Store charts. To do so, you'll need to meet high expectations for quality and functionality.

Three primary themes differentiate iOS from other platforms:

- **Clarity.** Throughout the system, text is legible at every size, icons are precise and lucid, adornments are subtle and appropriate, and a sharpened focus on functionality motivates the design. Negative space, color, fonts, graphics, and interface elements subtly highlight important content and convey interactivity.
- **Deference.** Fluid motion and a crisp, beautiful interface help people understand and interact with content while never competing with it. Content typically fills the entire screen, while translucency and blurring often hint at more. Minimal use of bezels, gradients, and drop shadows keep the interface light and airy, while ensuring that content is paramount.
- **Depth.** Distinct visual layers and realistic motion convey hierarchy, impart vitality, and facilitate understanding. Touch and discoverability heighten delight and enable access to functionality and additional content without losing context. Transitions provide a sense of depth as you navigate through content.

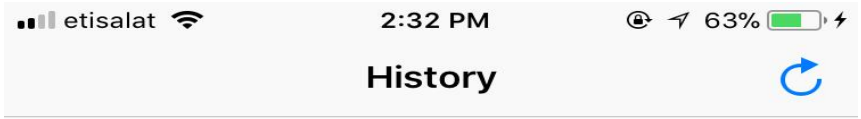
Color

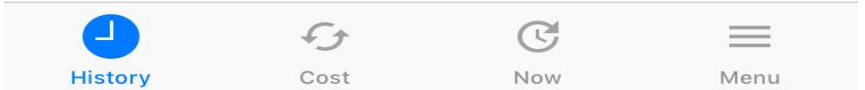
Color is a great way to impart vitality, provide visual continuity, communicate status information, give feedback in response to user actions, and help people visualize data. Look to the system's color scheme for guidance when picking app tint colors that look great individually and in combination, on both light and dark backgrounds.

Red	Orange	Yellow	Green	Teal Blue	Blue	Purple	Pink
R 255 G 59 B 48	R 255 G 149 B 0	R 255 G 204 B 0	R 76 G 217 B 100	R 90 G 200 B 250	R 0 G 122 B 255	R 88 G 86 B 214	R 255 G 45 B 85

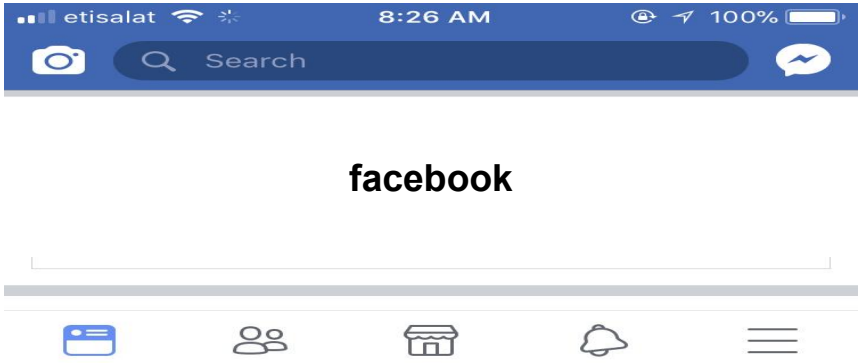
Navigation bar and Tab bar

A navigation bar appears at the top of an app screen, below the status bar, and enables navigation through a series of hierarchical screens. When a new screen is displayed, a back button, often labeled with the title of the previous screen, appears on the left side of the bar. Sometimes, the right side of a navigation bar contains a control, like an Edit or a Done button, for managing the content within the active view. A tab bar appears at the bottom of an app screen and provides the ability to quickly switch between different sections of an app.
Tab bars are translucent, may have a background tint, maintain the same height in all screen orientations, and are hidden when a keyboard is displayed. A tab bar may contain any number of tabs, but the number of visible tabs varies based on the device size and orientation.

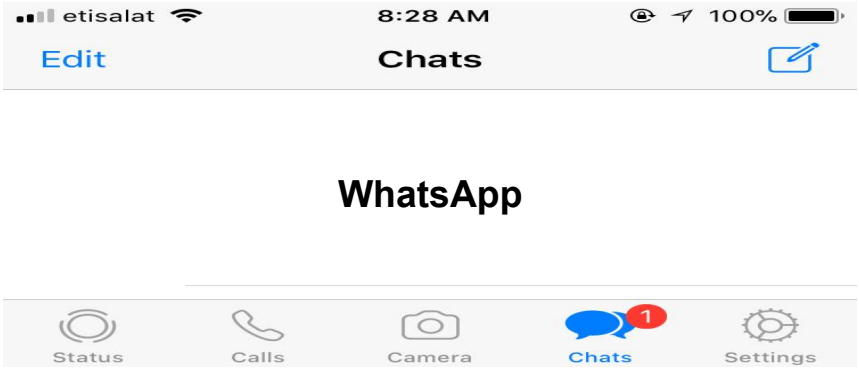




Examples



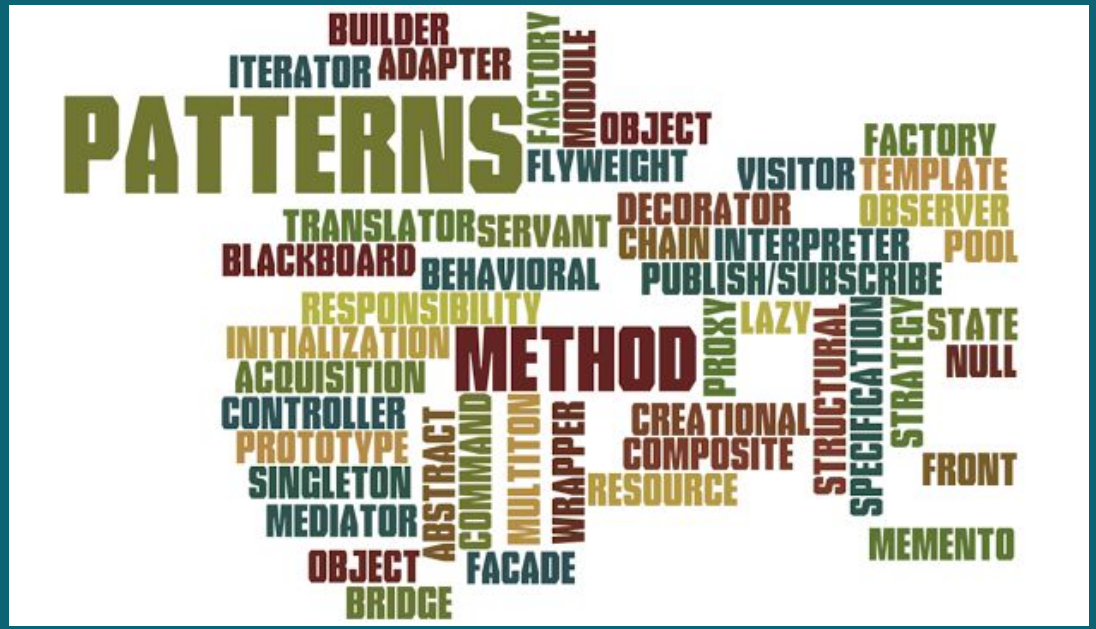
facebook



WhatsApp

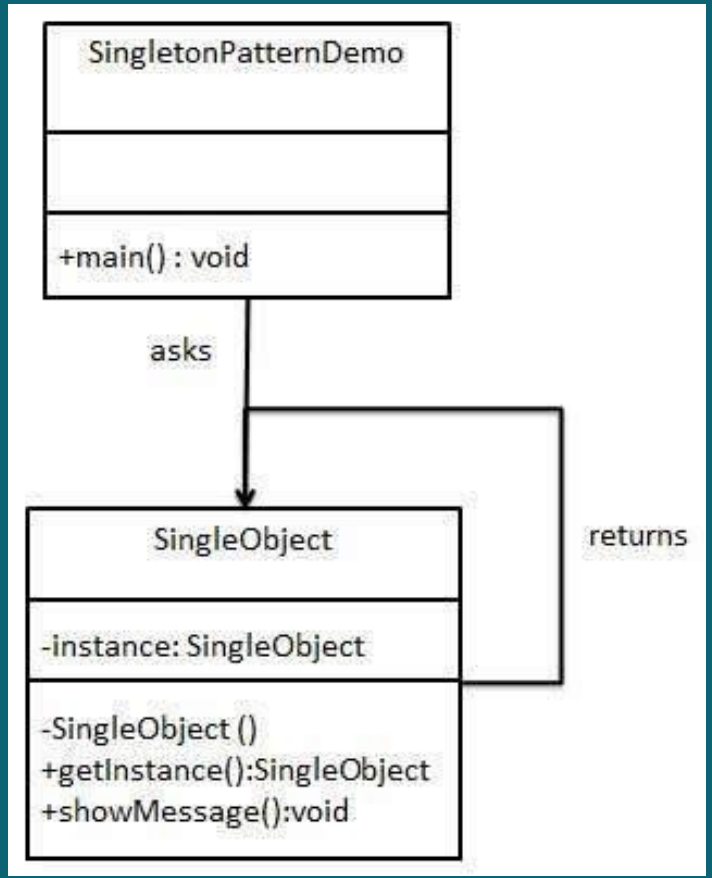
Design patterns

a general repeatable solution to a commonly occurring problem in software design. A design pattern is not a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations.



Singleton pattern:

we use Singleton pattern to ensure that there is only one instance of a class is created in the Java Virtual Machine. It is used to provide global point of access to the object.
We use singleton pattern in two main parts in our code:



Singleton pattern:

We use it in “ House” class as for sure we serve one house a time:

```
10  public class House {
11
12
13      private final List<PieEntry> pieEntries;
14
15      private static final House ourInstance = new House();
16      private final List<Entry> entries;
17      private final LineDataSet dataSet;
18      private float thisMonthReading;
19
20      public static House getInstance() {
21          return ourInstance;
22      }
23
24      private House() {
25          thisMonthReading = 0;
26          entries = new ArrayList<>();
27          pieEntries = new ArrayList<>();
28          dataSet = new LineDataSet(entries, "Watts");
29      }
30
31      public List<Entry> getEntries() {
32          return entries;
33      }
```

Singleton pattern:

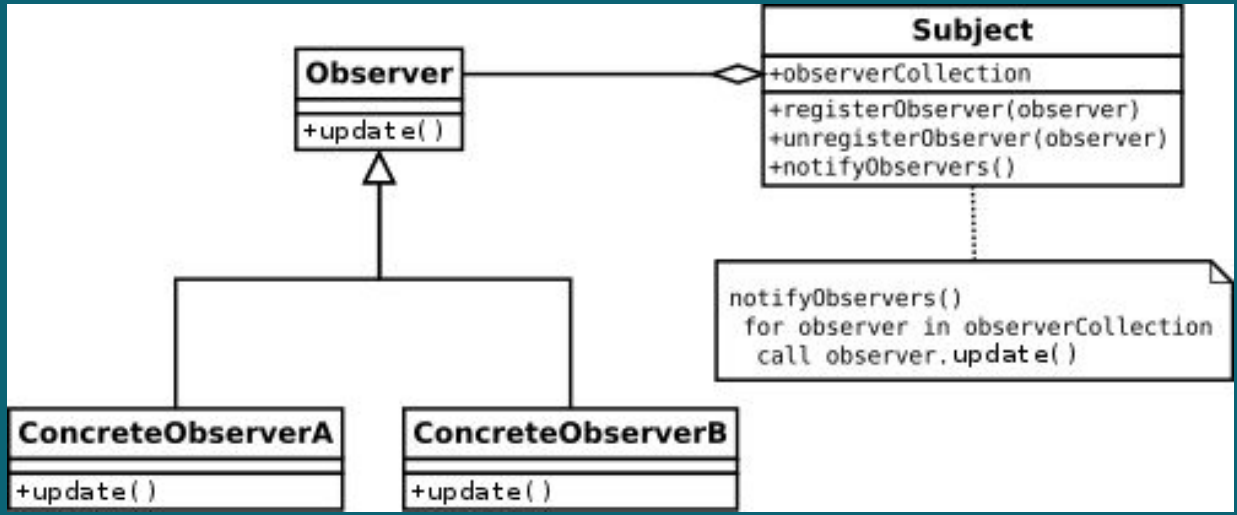
We also use it in “FirebaseUtils” the function which retrieve data from the cloud database to ensure that we retrieve readings just one time which helps us the improve the performance:

```
49      private FirebaseUtils() {
50          final FirebaseDatabase database = FirebaseDatabase.getInstance();
51          database.setPersistenceEnabled(true);
52          myRef = database.getReference();
53
54
```

```
250     }
251
252     static FirebaseUtils getInstance() {
253         return ourInstance;
254     }
```

Observer pattern:

The observer pattern is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods, we use it to update our fragment views while keeping the principle of loose coupling between objects that interact with each other. It allows sending data to other objects effectively without any change in the Subject or Observer classes.



Observer pattern:

We use it in “ Rooms ”

```
//Registering observer
FirebaseUtils.getInstance().addObserver( o: this);
update( observable: null,  o: null);
```

Also we use it in readings to update charts

```
//Implementation for the Observer interface to update the fragment views
@Override
public void update(Observable observable, Object o) {
    latestReadingTextView.setText(String.format(Locale.US,  format: "%.2g", FirebaseUtils.getInstance().getTotalLatestReading()))
    ;
    totalReadingTextView.setText(String.format(Locale.US,  format: "%.4g", FirebaseUtils.getInstance().getTotalReading()));
    mAdapter.notifyDataSetChanged();
}
}
```

04- Machine learning

Why Machine Learning ?

- More intelligence
- From monitoring/manual controlling to reducing bills..
- Trending

Algorithm

We aim to recommend a certain behavior by which users could reduce their bills. So, we have two options to make a recommender system.

1- Regression.

2- Deep Learning.

Regression

Regression VS Deep Learning

Linear Regression

1. Data Acquisition and preprocessing
    a. Creating Dataset
    b. Feature scaling and mean normalization
2. Model Representation
    a. Hypothesis
    b. Cost function
3. Parameters Learning
    c. Gradient Descent
3. Prediction

Data Acquisition and preprocessing

a. **Dataset**

usage	priority	budget	hours
116.71	2	300	5
129.8	1	601	14
345.4	1	3790	13
292.6	1	2346	19
288.88	3	1434	8
224.4	4	252	8
129.8	4	3140	20
184.8	3	50	1
154	4	4057	6

Data Acquisition and preprocessing

B. Feature Scaling and Mean Normalization:

Used to speed up the operation of Gradient Descent (getting optimum Thetas).

Where:

- X(i) : entries of the dataset.
- Mu : the mean of dataset.
- S (sigma) : the standard deviation of dataset.

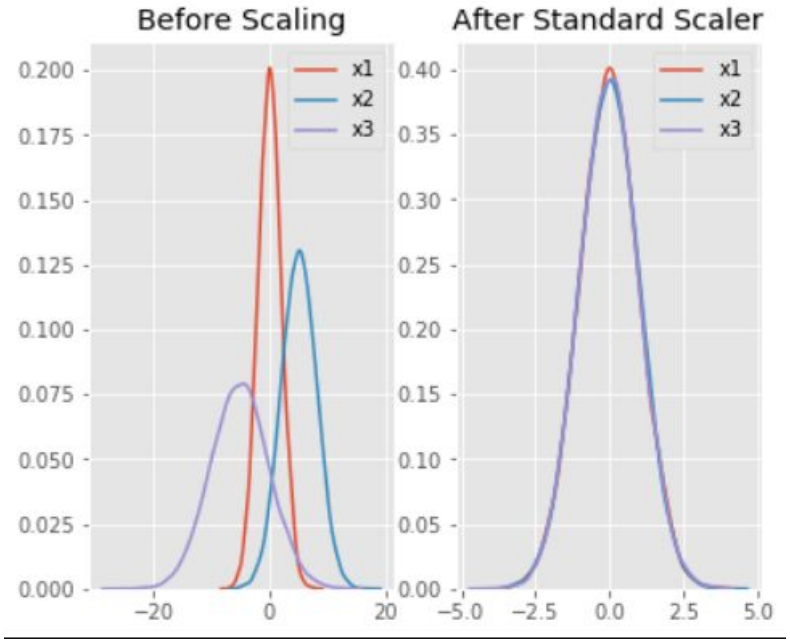
$$x_i := \frac{x_i - \mu_i}{s_i}$$

Data Acquisition and preprocessing

B. Feature Scaling and Mean Normalization.

.

$$x_i := \frac{x_i - \mu_i}{s_i}$$



Model Representation

a. Hypothesis:

$$hi(x) = theta\_0 + theta\_1 * x1 + theta\_2 * x2 + theta\_3 * x3$$

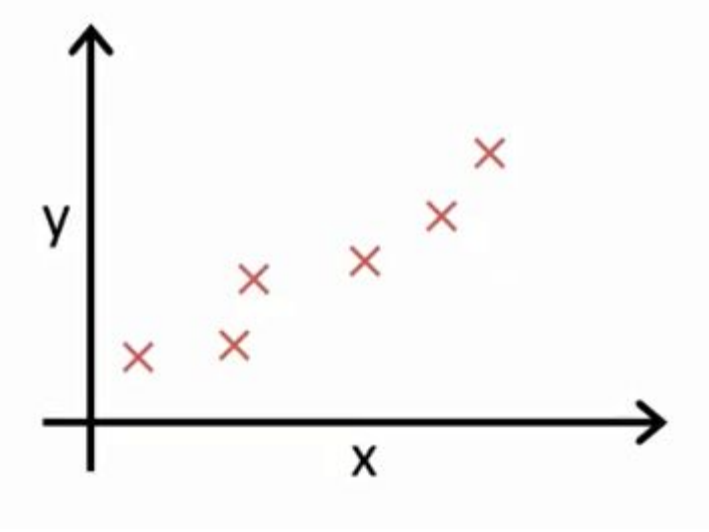
Where:

hi(x): predicted output (number of hours recommended for circuit breaker i).
Theta_n : prediction weights.
i: index (1,2,….etc).
x1 : Priority of circuit breaker i.
x2 : determined budget of overall consumption.
x3 : average consumption of circuit breaker i.

Model Representation

a. Hypothesis:

$$hi(x) = theta\_0 + theta\_1 * x1 + theta\_2 * x2 + theta\_3 * x3$$

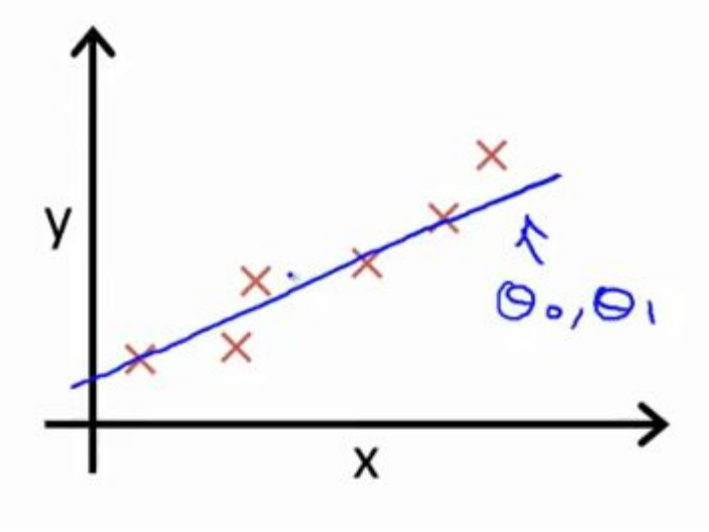


Model Representation

a. Hypothesis:

$$hi(x) = theta\_0 + theta\_1 * x1 + theta\_2 * x2 + theta\_3 * x3$$

h(x)=theta_0 +
theta_1 * X



Model Representation

b. Cost function :

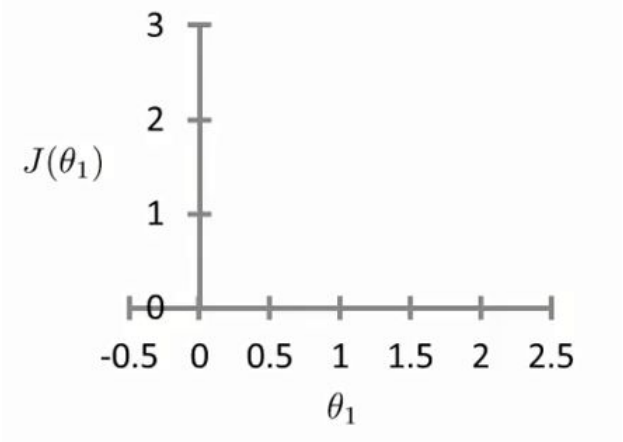
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

Where:
J(theta) : the accuracy of our prediction.
M : total number of entries in the training set.
Yi : actual output at index i.

Model Representation

b. Cost function :

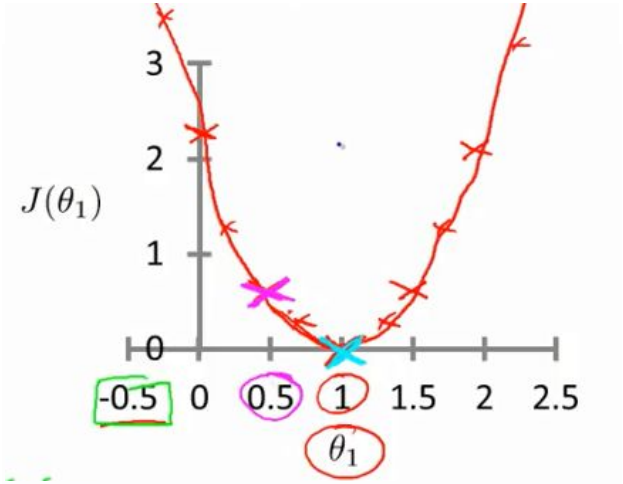
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$



Model Representation

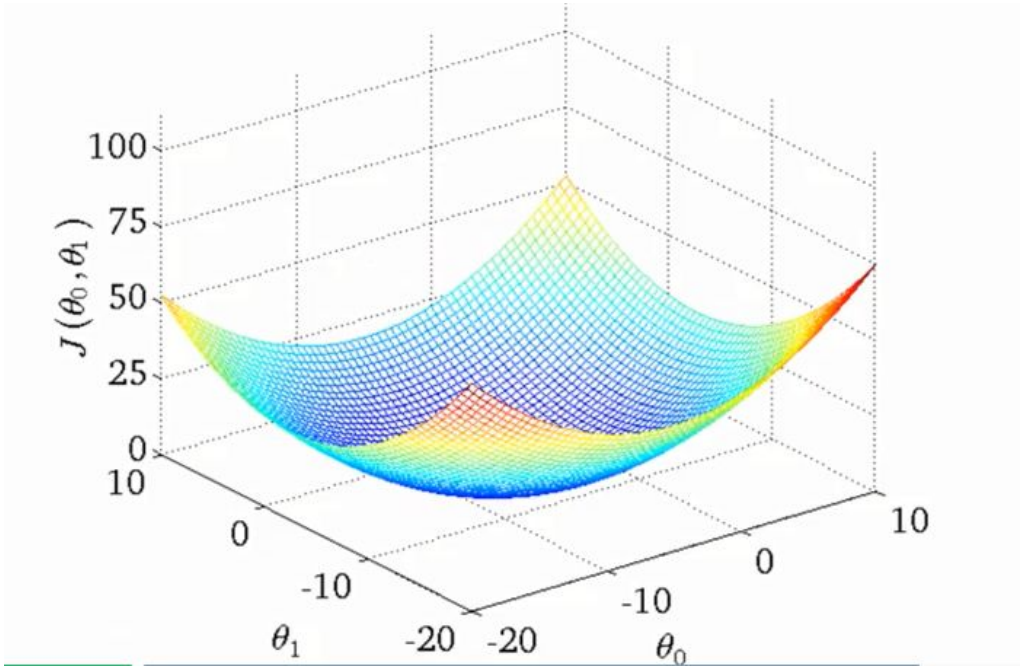
b. Cost function :

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$



Model Representation

b. Cost function with multiple features :



Model Representation

b. Cost function :

Features: Priority, Usage, Budget.

Hypothesis : hi(x) = theta_0 + theta_1 * x1 + theta_2 * x2 + theta_3 * x3.

Parameters : theta_0, theta_1, theta_2, theta_3.

Cost Function : $$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

Goal : minimizing J(theta) - Cost Function -.

Parameters Learning

c. Gradient Descent :

We need to estimate the parameters in the hypothesis function so The gradient descent algorithm is:

1. Start with random theta values.
2. Keep changing thetas to reduce the cost function until we minimize it.

Parameters Learning

c. Gradient Descent :

We need to estimate the parameters in the hypothesis function so The gradient descent algorithm is:

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$

$\ldots$

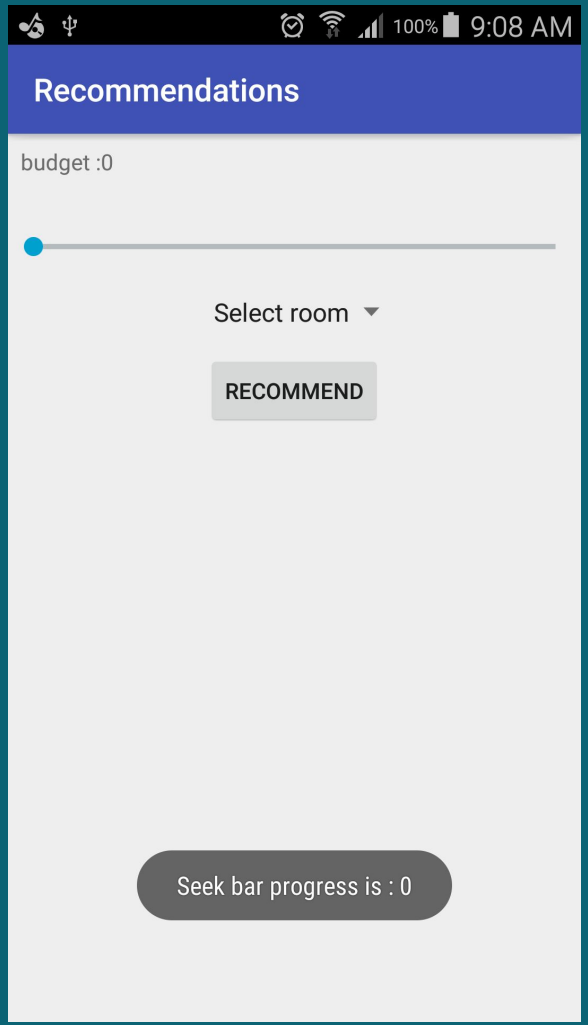
}

Where:

- M : # training set entries.
- Alpha : Learning rate.

Prediction

Recommendation Screen



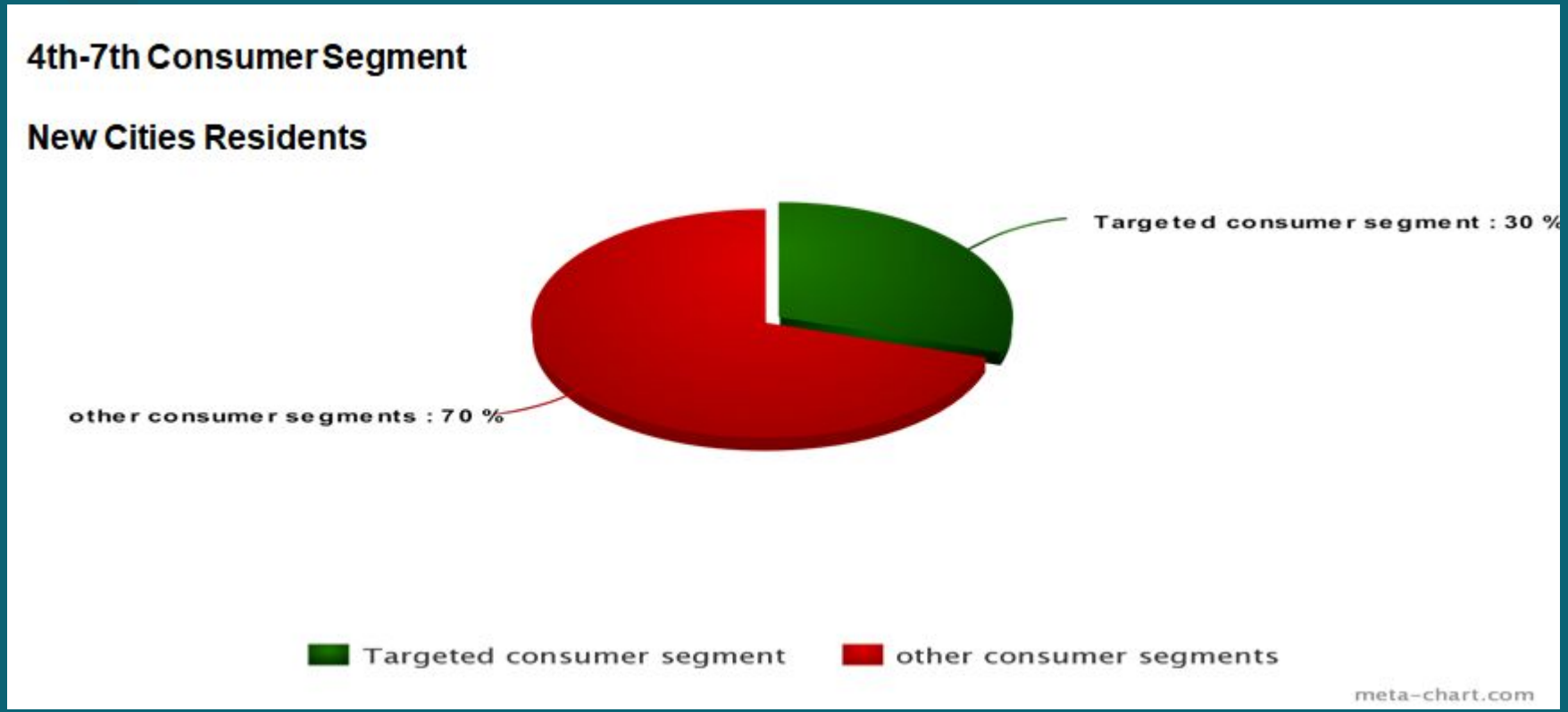
Outlines

1. Introduction
2. Proposed Solution
   - A. Project overview
   - B. System features
   - C. System Architecture
3. System modules
   - A. Hardware module
   - B. Software module
     - Android Application
     - IOS Application
   - C. Database server
   - D. Machine learning
4. Marketing
5. Time plan

Market size:

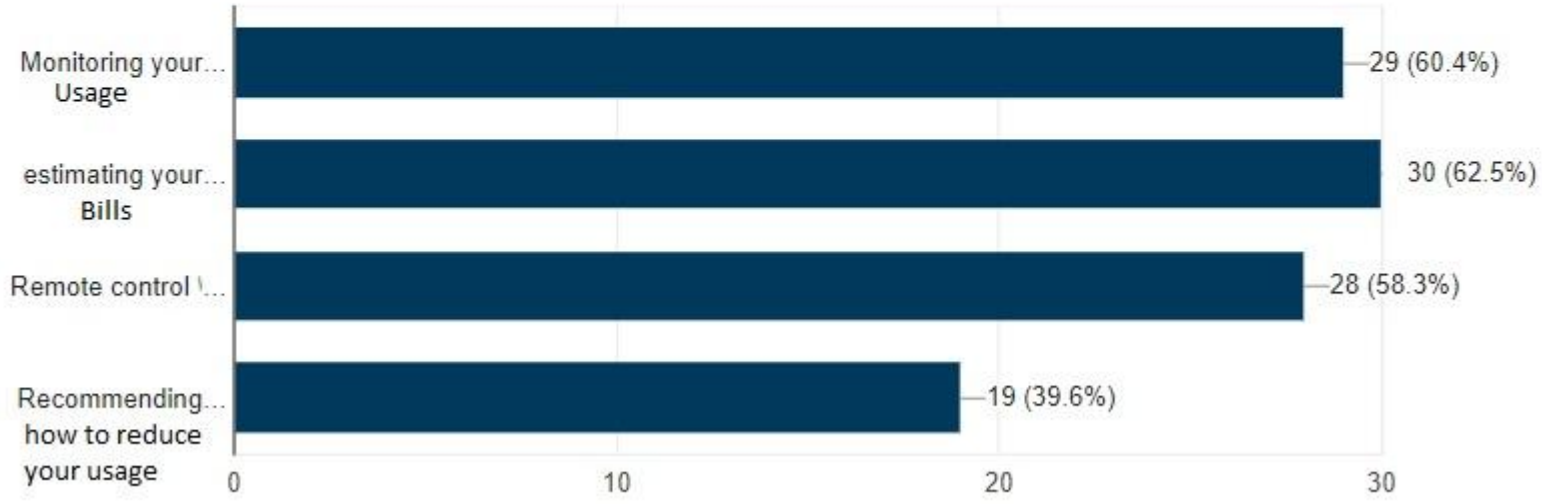
Egypt Population (2016)	95,688,681
Access to Electricity	99.80%
Residents Usage	21.31%
New Cities Full Capacity	4,350,000

Market Segmentation:



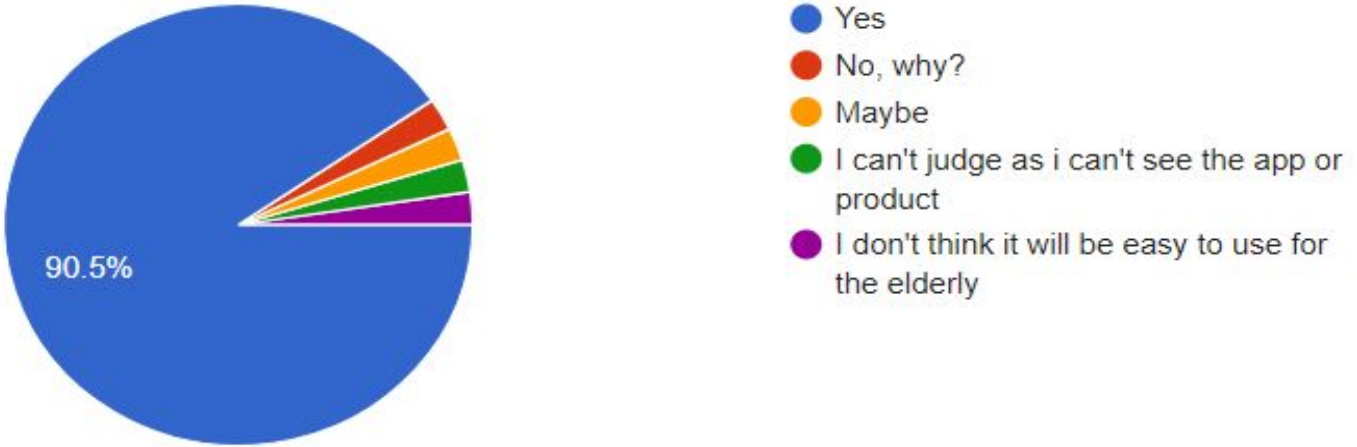
Survey results

Please tick against the most important features for you to use this product effectively



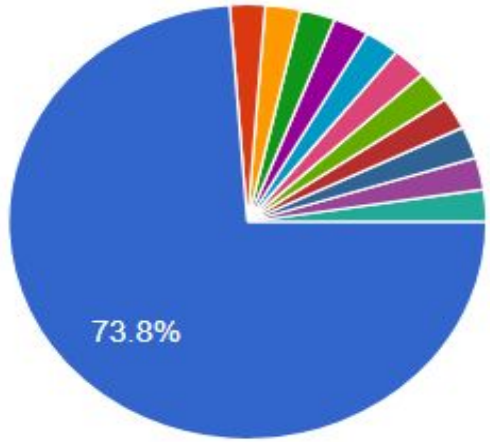
Survey results

Do you find it easy to use a mobile app that is connected to a hardware device? هل تري ان هذا المنتج سهل الاستعمال؟



Survey results

If this product is available in the market, will you consider purchasing it?
اذا تواجد هذا المنتج في السوق، هل ستنوي شراؤه؟



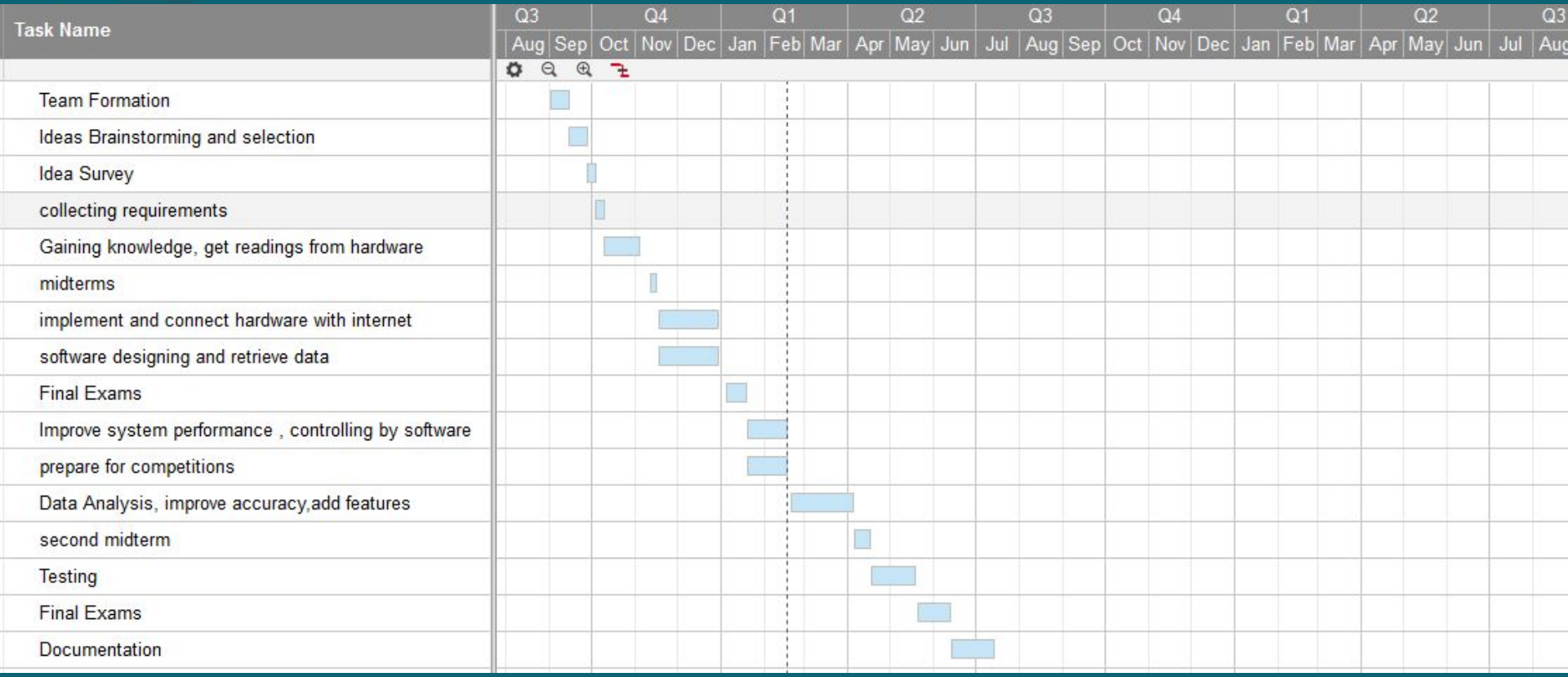
- YES
- No, why?
- Maybe
- انه فعلا قلل فوا reviews حسب السعر سراحه و...
- لا اعرف
- although it's an amazing idea but in...
- سوف اشتريه إذا كانت تكلفته متوسطة
- معرفة السعر اولا 😂

1/2

Timeline plan

Task Name	Duration	Start	Finish
Team Formation	10d	09/01/17	09/14/17
Ideas Brainstorming and selection	10d	09/14/17	09/27/17
Idea Survey	5d	09/27/17	10/03/17
collecting requirements	5d	10/03/17	10/09/17
Gaining knowledge, get readings from hardware	20d	10/09/17	11/03/17
midterms	4d	11/11/17	11/15/17
implement and connect hardware with internet	31d	11/17/17	12/29/17
software designing and retrieve data	31d	11/17/17	12/29/17
Final Exams	11d	01/04/18	01/18/18
Improve system performance , controlling by software	21d	01/19/18	02/16/18
prepare for competitions	21d	01/19/18	02/16/18
Data Analysis, improve accuracy,add features	33d	02/19/18	04/04/18
second midterm	8d	04/05/18	04/16/18
Testing	24d	04/17/18	05/18/18
Final Exams	18d	05/20/18	06/12/18
Documentation	23d	06/13/18	07/13/18

Timeline plan




Q&A

THANK YOU!