A System for Automated Notification and Monitoring for an Empty Room with Lights On

by

Mohammad Shafat (011480430)

Purva Patel (011325002)

Amrapali Sarkar (010804599)

Praveen Elagudri (011448463)

DEPARTMENT OF ENGINEERING

SAN JOSÉ STATE UNIVERSITY

MAY 2017

Abstract

Mobile Sensor services have changed numerous parts of mobile and cloud computing with regards to Internet of Things. Our project provides a low cost-effective, flexible control and monitoring system with the aid of an Arduino microcontroller and node server with IP connectivity for access to and control of equipment and devices remotely using Android-based smartphone app. The proposed system does not require a dedicated server PC with respect to similar systems and offers a new communication protocol for monitoring and controlling the home environment with more than just switching functionality. The smart interfaces and device definitions to ensure interoperability between android devices from various manufacturers of electrical equipment, meters and Smart Energy enables products to allow manufactured. We introduced the proposed energy control systems design intelligent services for users and provides, we show their implementation, with smartphone.

Table of Contents

List of Tables:

List of Figures

1. INTRODUCTION

Intelligent management of the power system, facilitate the joint use of the current and minimizes power loss during transmission and power consumption is highlighted by the global community, academic institutions, and State administration. Environment monitoring and device control allows new level of comfort in colleges or homes and it can also manage the energy consumption efficiently which in turns promotes the saving. Remote controlling of the devices offers many advantages to senior citizens and people with disabilities which helps them in being more autonomous and increasing quality of life. We have proposed a system that develops a Smart Control and Monitoring System by harnessing the power of IoTs at low-cost which provides flexible and scalable architecture for university automation. It will provide security, energy efficiency.

We propose to design and implement a system for automated notifications and real time monitoring of a lighted empty room. We implement the proposed system and develop related hardware and software. We are collaborating IoT, Cloud computing and Mobile Application Development. Our system will help the user to get notified about empty rooms with lights ON.

2. LITERATURE REVIEW

Exploiting features of Android mobile devices for saving energy applications which uses techniques like Arduino microcontroller. In this project by using Android application user can control energy systems like lights. After that control system based on Android smartphone introduced in 2014, preferred Techniques like PIC Controller. A user logs into the smartphone interface, and clicks the buttons gently to send message commands which will be transmitted to home information Centre through the GSM network. Then the PIC processor recognizes the specified command, and controls the home appliance switches in the wireless radio frequency manner to achieve remote control of appliances ultimately.

3. REQUIREMENTS

Some of the requirements to run our application are

- Android phone with version Jelly Bean or above.
- An Empty room with lights on.
- Internet connectivity
- Arduino Uno
- Temperature Sensor
- Motion Sensor
- LED
- Light Sensor

4. MOBILE UI DESIGN WIREFRAMES AND STORYBOARDS

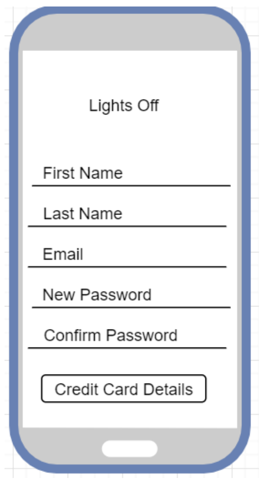


Figure 1. Registration Screen

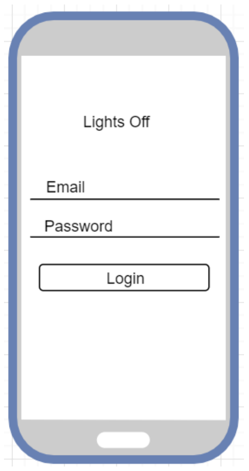


Figure 2. Login Screen

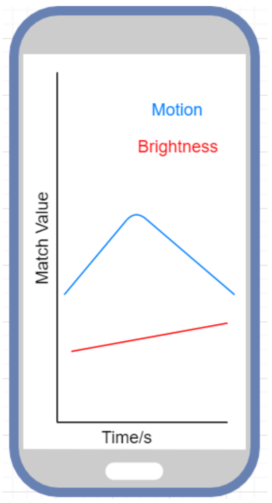


Figure 3. Charts Screen

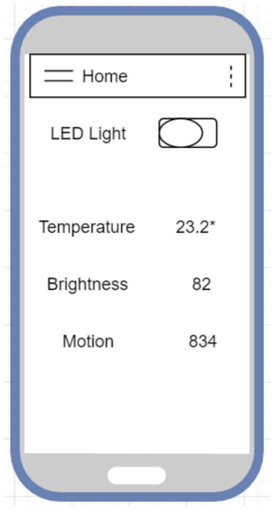


Figure 4. Dashboard

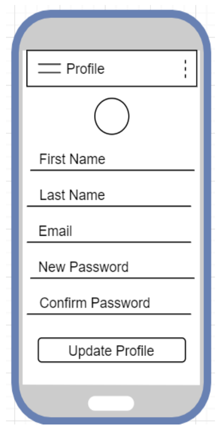


Figure 5. Update Profile Screen

5. HIGH LEVEL ARCHITECTURE DESIGN

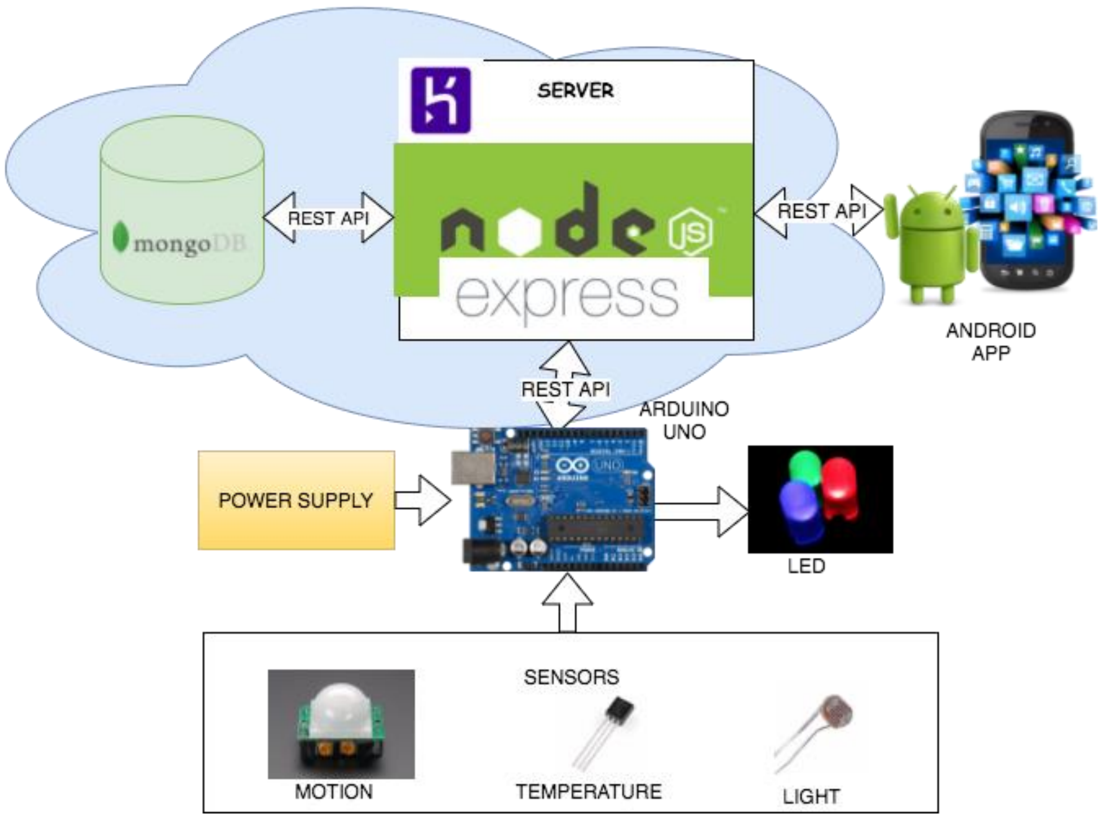


Figure 6. Project Architecture

5.1 Software Design

Technologies used in the software of the proposed architecture is as follows:

Layers in Architecture	**Technology used**
Front End	Android
Backend Server	Node.js and Express.js
Backend Deployment	Heroku
Database	MongoDB

Table 1 : Technologies used

Android Application: The android application is built on OS version 4.2, Jelly bean. It is supported by 95% of the Android devices as per May, 2017. This app connects to the device which is kept in a room to detect its emptiness with switched on lights. The app is made secured with Tokenization authentication model. The user first installs the app and registers the device with his email and password. Each time the user opens the app he must login into the app with his email and password. The credentials are checked in the database via the backend server and a token is generated. The user is now logged in using this token and sees the dashboard.

Using this app, the user can monitor the data in near real time. He can switch on and off the LED light easily and quickly from the mobile devices via a simple and comfortable GUI application. The system then acts and respond to these commands by taking actions per commands and gives the result to the user. The user gets a notification whenever the device detects an empty room with light on. The user can also see the result on Android mobile application from anywhere.

6. COMPONENT LEVEL DESIGN

6.1 IoT:

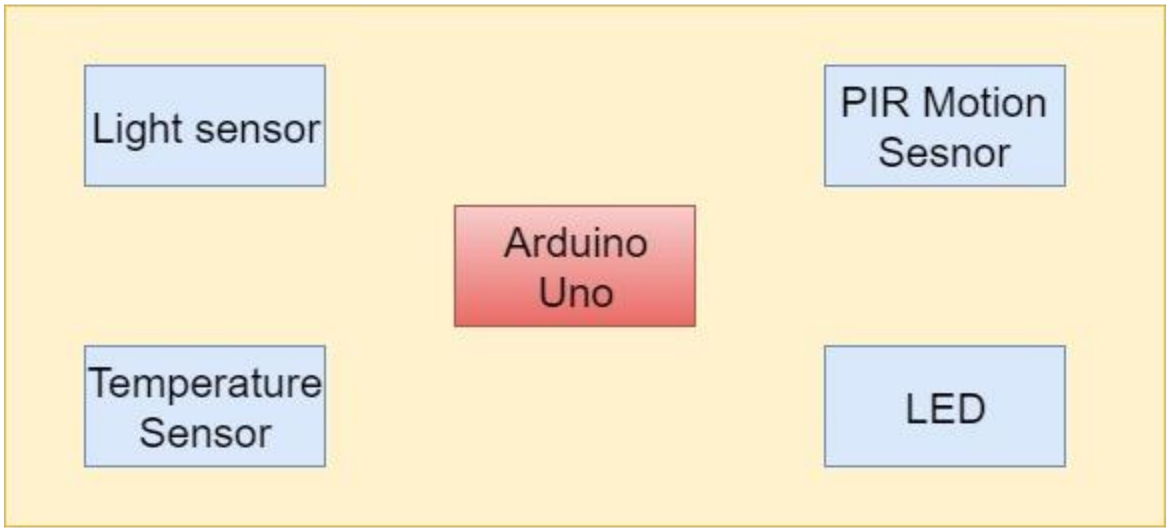


Figure 7. IoT Architecture

This component consists a few sensors like Light sensor, Temperature sensor, PIR Motion Sensors and LED.

6.1.1 Light sensor:

A Light Sensor is something that a robot can use to detect the current ambient light level - i.e. how bright/dark it is. There are a range of different types of light sensors, including 'Photo resistors', 'Photodiodes', and 'Phototransistors'. The sensor included in the BOE Shield-Bot kit, and the one we will be using, is called a Phototransistor. A photoresist or operates similarly to a phototransistor however it changes its resistance based on the amount of light that falls upon it. Photo resistors tend to be less sensitive, also.

6.1.2 PIR Motion sensor:

PIR sensors are more complicated than many of the other sensors explained in these tutorials (like photocells, FSRs and tilt switches) because there are multiple variables that affect the sensors input and output. To begin explaining how a basic sensor works, we'll use this rather nice diagram

The PIR sensor itself has two slots in it, each slot is made of a special material that is sensitive to IR. The lens used here is not doing much and so we see that the two slots can 'see' out past some distance (basically the sensitivity of the sensor). When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors. When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves. When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.

6.1.3 Temperature sensor:

Temperature Sensors measure the amount of heat energy or even coldness that is generated by an object or system, allowing us to “sense” or detect any physical change to that temperature producing either an analogue or digital output. There are many different types of Temperature Sensor available and all have different characteristics depending upon their actual application.

6.1.4 Led Sensor:

A light-emitting diode (LED) is a two-lead semiconductor light source. It is a p–n junction diode that emits light when activated. When a suitable voltage is applied to the leads, electrons are able to recombine with electron holes within the device, releasing energy in the form of photons. This effect is called electroluminescence, and the color of the light (corresponding to the energy of the photon) is determined by the energy band gap of the semiconductor.

6.2 Database:

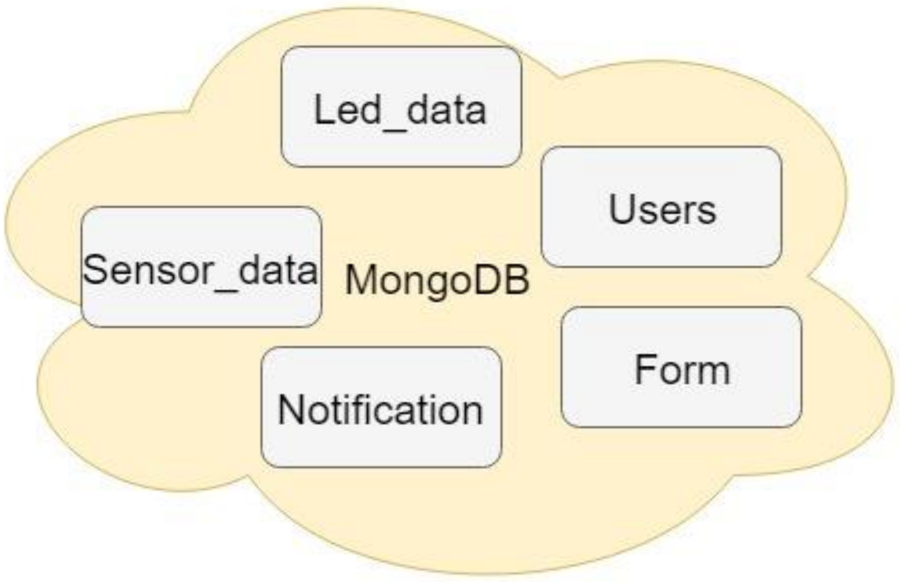


Figure 8. Database Architecture

Database used in this project is a NoSql database. **MongoDB** (from *hu**mongo**us*) is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemas. MongoDB is developed by MongoDB Inc. and is free and open-source, published under a combination of the GNU Affero General Public License and the Apache License.

The Following are the collections used in our Database:

- Users
- Form
- Notification
- Led_Data

- Sensor_data

6.3 Backend:

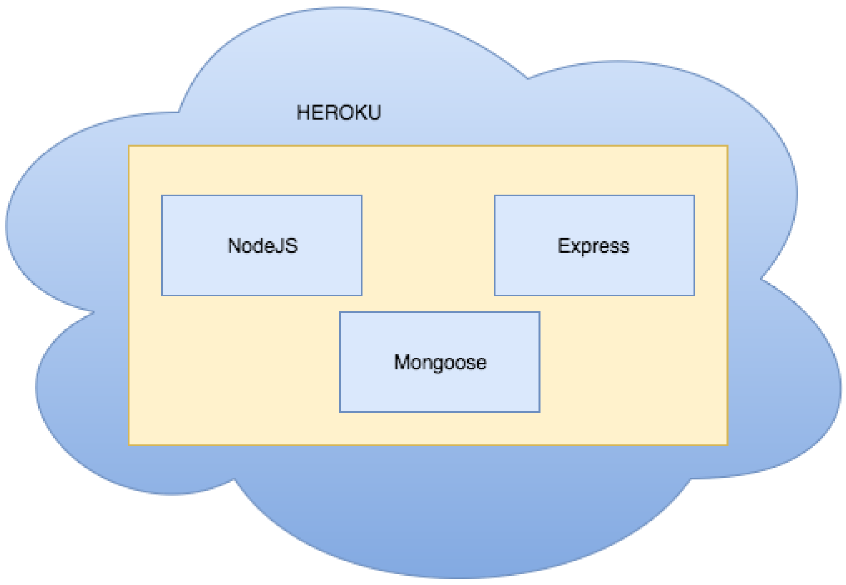


Figure 9. Backend architecture

Backend of the project is written on Node.js and Express.js. Mongoose library is used to connect the backend to the database. The Backend id deployed on Heroku. Heroku is Platform as a Service which takes care of build and deployment.

6.4 Android:

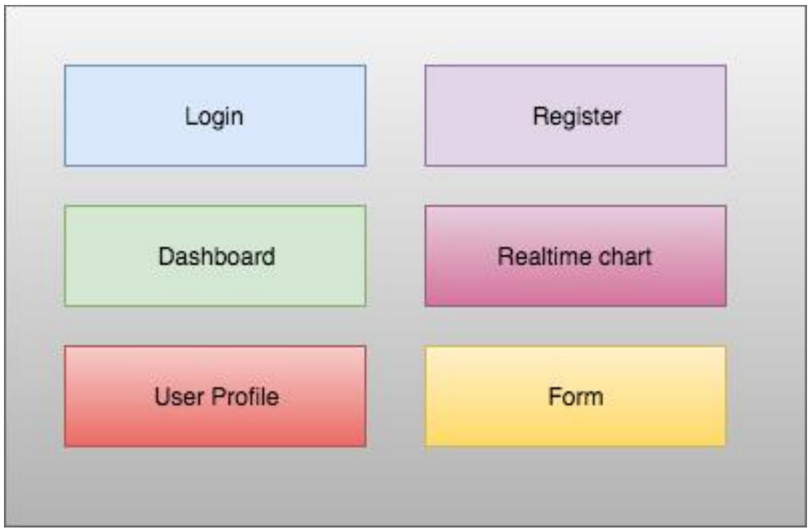


Figure 10. Android App Components

The following components are used in the Android application:

Login: The user can login to the dashboard using his email id and password.

Register: If the user is not registered in the database then he can register himself using this component.

Dashboard: This the main dashboard where the user can see the real time values of the sesnors and control the LED on and off

Real Time chart: User can see the real time line chart of the sensor values

User Profile: User can see his details and update them.

Form: Its a survey form to know more about the product for future purposes.

7. USER NAVIGATION SEQUENCE

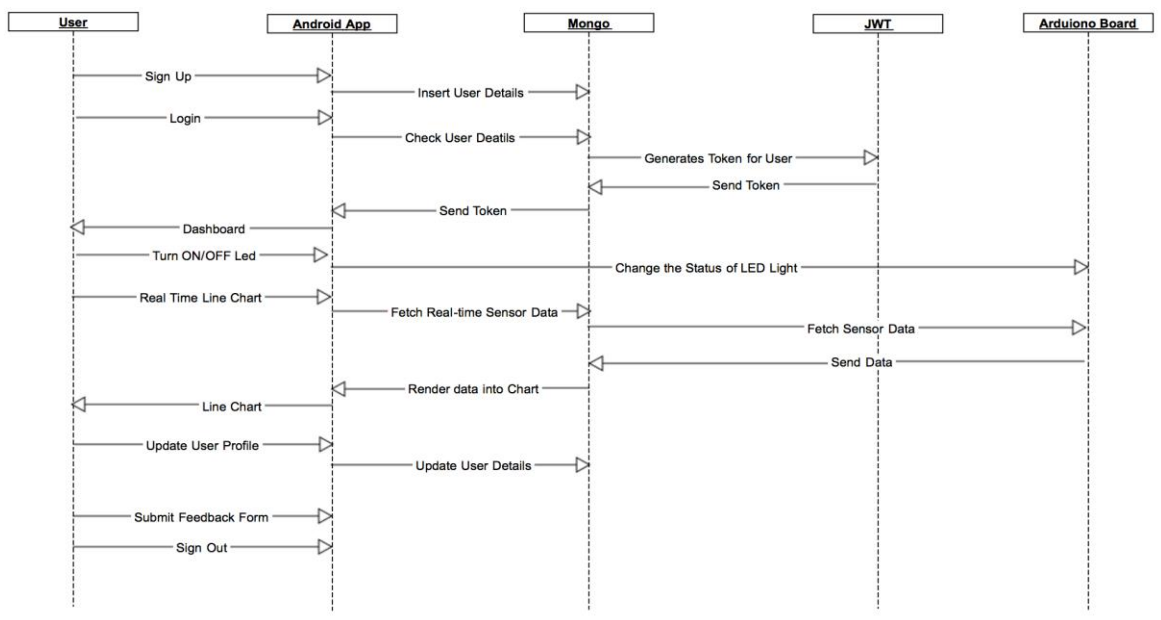


Figure 11: User Navigation Sequence Diagram

8. MOBILE & CLOUD TECHNOLOGIES

8.1 Mobile Technologies:

The following technologies are used in Mobile app development:

- Navigation Drawer
- Retrofit
- Graphview
- Fragments
- Intents
- Service
- Asynctask

8.1.1 Navigation Drawer:

The drawer slides on top of the substance rather than the substance sliding endlessly. With this, one can get to any top level substance from anyplace in the application and enables you to make a compliment route structure. It likewise has an activity bar at the top. Bring down adaptations likewise bolster it through support library.

8.1.2 Retrofit:

Retrofit is a type-safe REST client for Android (or just Java) developed by Square. The library provides a powerful framework for authenticating and interacting with APIs and sending network requests with OkHttp. See this guide to understand how OkHttp works. This library makes downloading JSON or XML data from a web API fairly straightforward. Once the data is downloaded then it is parsed into a Plain Old Java Object (POJO) which must be defined for each "resource" in the response.

8.1.3 Graphview:

GraphView is a library for Android to programmatically create flexible and nice-looking diagrams. It is **easy** to understand, to integrate and to customize. Create Line Graphs, Bar Graphs, Point Graphs or implement your own custom types.

8.1.4 Fragment:

A Fragment represents a behavior or a portion of user interface in an Activity. You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities. You can think of a fragment as a modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running (sort of like a "sub activity" that you can reuse in different activities).

A fragment must always be embedded in an activity and the fragment's lifecycle is directly affected by the host activity's lifecycle. For example, when the activity is paused, so are all fragments in it, and when the activity is destroyed, so are all fragments. However, while an activity is running (it is in the resumed lifecycle state), you can manipulate each fragment independently, such as add or remove them. When you perform such a fragment transaction, you can also add it to a back stack that's managed by the activity—each back stack entry in the activity is a record of the fragment transaction that occurred. The back stack allows the user to reverse a fragment transaction (navigate backwards), by pressing the Back button.

When you add a fragment as a part of your activity layout, it lives in a ViewGroup inside the activity's view hierarchy and the fragment defines its own view layout. You can insert a fragment into your activity layout by declaring the fragment in the activity's layout file, as a <fragment> element, or from your application code by adding it to an existing ViewGroup. However, a fragment is not required to be a part of the activity layout; you may also use a fragment without its own UI as an invisible worker for the activity.

8.1.5 Intent:

An Intent is a messaging object you can use to request an action from another app

component. Although intents facilitate communication between components in several ways, there

are three fundamental use cases:

- Starting an activity:An Activity represents a single screen in an app. You can start a new instance of an Activity by passing an Intent to startActivity(). The Intent describes the activity to start and carries any necessary data.If you want to receive a result from the activity when it finishes, call startActivityForResult(). Your activity receives the result as a separate Intent object in your activity's onActivityResult() callback. For more information, see the Activities guide.

- Starting a service: A Service is a component that performs operations in the background without a user interface. With Android 5.0 (API level 21) and later, you can start a service with JobScheduler. For more information about JobScheduler, see its API-reference documentation. For versions earlier than Android 5.0 (API level 21), you can start a service by using methods of the Service class. You can start a service to perform a one-time operation (such as downloading a file) by passing an Intent to startService(). The Intent describes the service to start and carries any necessary data. If the service is designed with a client-server interface, you can bind to the service from another component by passing an Intent to bindService(). For more information, see the Services guide.
- Delivering a broadcast: A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging. You can deliver a broadcast to other apps by passing an Intent to sendBroadcast() or sendOrderedBroadcast().

8.1.6 Service:

A Service is an application component that can perform long-running operations in the background, and it does not provide a user interface. Another application component can start a service, and it continues to run in the background even if the user switches to another application. Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC). For example, a service can handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

8.1.7 Asynctask:

AsyncTask enables proper and easy use of the UI thread. This class allows you to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers. AsyncTask is designed to be a helper class around Thread and Handler and does not constitute a generic threading framework. AsyncTasks should ideally be used for short operations (a few seconds at the most.) If you need to keep threads running for long periods of time, it is highly recommended you use the various APIs provided by the java.util.concurrent package such as Executor, ThreadPoolExecutor and FutureTask. An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. An asynchronous task is defined by 3 generic types, called Params, Progress and Result, and 4 steps, called onPreExecute, doInBackground, onProgressUpdate and onPostExecute.

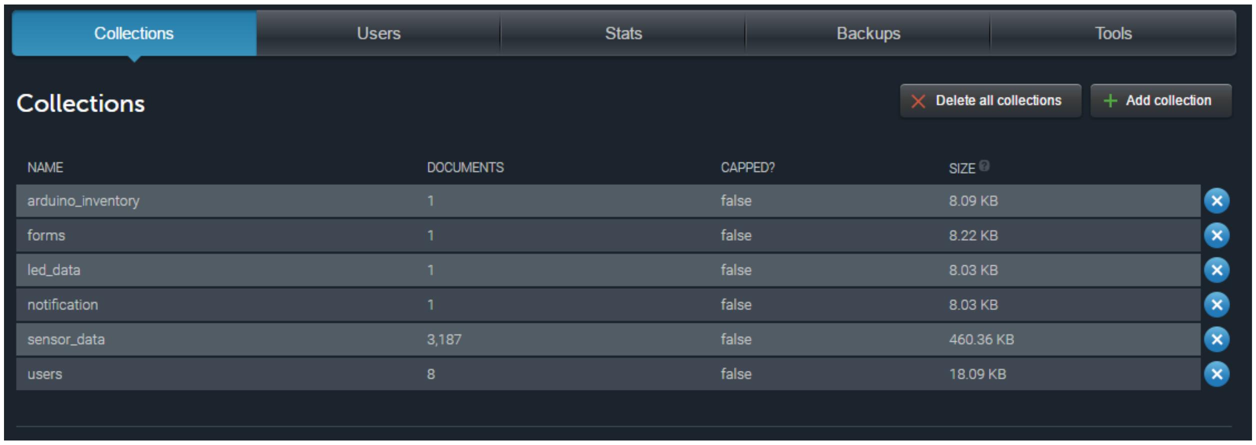
8.2 Cloud Technologies:

The following cloud technologies are used for deployment.

- Mlab
- Heroku

8.2.1 Mlab:

MLab is a fully managed cloud database service that hosts MongoDB databases. mLab runs on cloud providers Amazon, Google, and Microsoft Azure, and has partnered with platform-as-a-service providers.

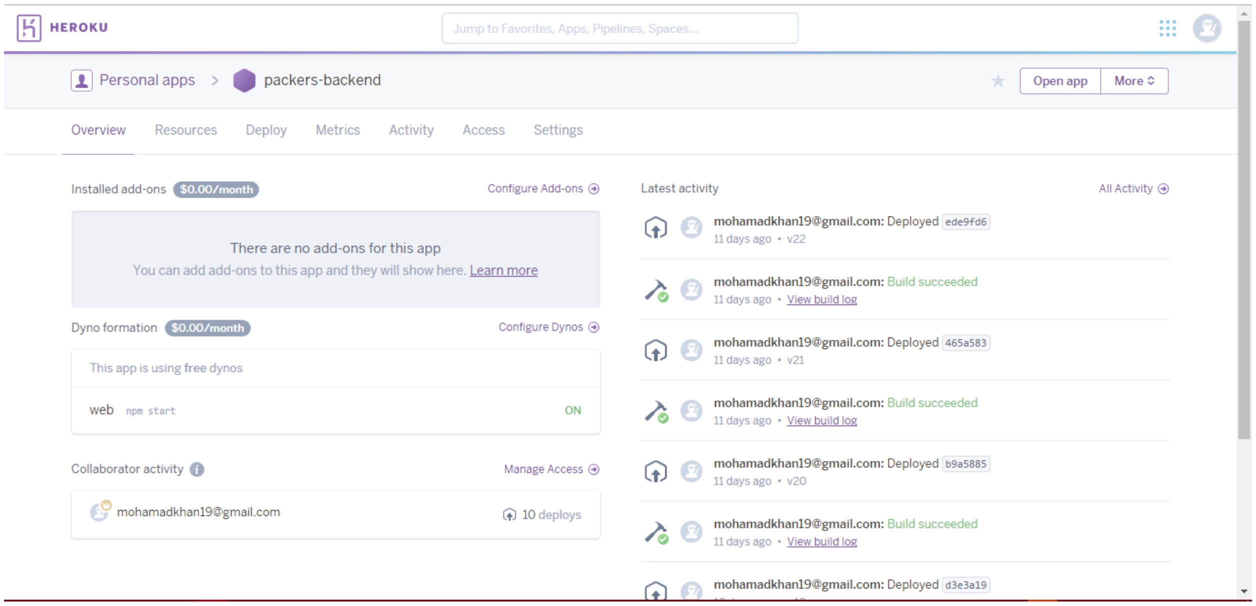


NAME	DOCUMENTS	CAPPED?	SIZE
arduino_inventory	1	false	8.09 KB
forms	1	false	8.22 KB
led_data	1	false	8.03 KB
notification	1	false	8.03 KB
sensor_data	3,187	false	460.36 KB
users	8	false	18.09 KB

Figure 12. Database Structure

8.2.2 Heroku:

Heroku is a cloud Platform-as-a-Service (PaaS) supporting several programming languages that is used as a web application deployment model. Heroku, one of the first cloud platforms. It supports Java, Node.js, Scala, Clojure, Python, PHP, and Go. For this reason, Heroku is said to be a polyglot platform as it lets the developer build, run and scale applications in a similar manner across all the languages.



Installed add-ons $0.00/month

There are no add-ons for this app
You can add add-ons to this app and they will show here. Learn more

Dyno formation $0.00/month

This app is using free dynos

web `npm start` ON

Collaborator activity

mohamadkhan19@gmail.com 10 deploys

Latest activity

mohamadkhan19@gmail.com: Deployed ede9fd6
11 days ago • v22

mohamadkhan19@gmail.com: Build succeeded
11 days ago • View build log

mohamadkhan19@gmail.com: Deployed 465a583
11 days ago • v21

mohamadkhan19@gmail.com: Build succeeded
11 days ago • View build log

mohamadkhan19@gmail.com: Deployed b9a5885
11 days ago • v20

mohamadkhan19@gmail.com: Build succeeded
11 days ago • View build log

mohamadkhan19@gmail.com: Deployed d3e3a19

Figure 13:Heroku

9. RESTFUL AND SERVER SIDE DESIGN

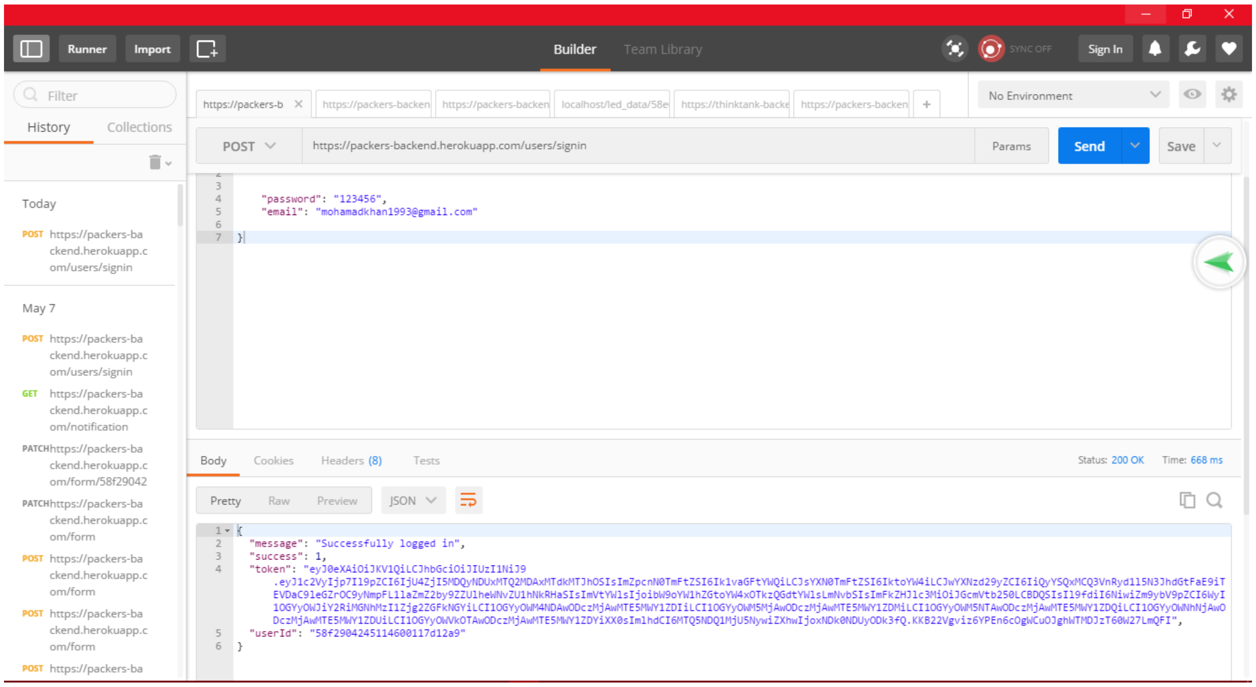
Web Server is the link between database and IoT and database and Android App. It consists of REST APIs. The REST APIs are exposed to the outer world. Only authenticated users can access the data from the database. In order to authenticate the users we are using tokenization methodology. Whenever the user signin, a Token is generated. Token is generated via JSON Web Token (JWT) library.

The following table consist of the uri, endpoints , methods and body.

URI	Endpoint	Method	Body
https://packers-backend.herokuapp.com	**/user/signup**	**Post**	**{ Name, Address, Email, Password }**
https://packers-backend.herokuapp.com	**/user/signin**	**Post**	**{ Email, Password }**
https://packers-backend.herokuapp.com	**/user/:id**	**Delete**	**{}**
https://packers-backend.herokuapp.com	**/sensor_data**	**Get**	**{}**
https://packers-backend.herokuapp.com	**/sensor_data**	**Post**	**{ Timestamp, Temperature, Motion, Brightness }**

https://packers-backend.herokuapp.com	**/notification**	**Get**	**{}**
https://packers-backend.herokuapp.com	**/notification/:id**	**Patch**	**{ notification }**
https://packers-backend.herokuapp.com	**/led_data**	**Get**	**{}**
https://packers-backend.herokuapp.com	**/led_data**	**Put**	**{ Led }**
https://packers-backend.herokuapp.com	**/latest_sensor_data**	**Get**	**{}**

Table 2: REST APIs


Runner
Import
Builder
Team Library
SYNC OFF
Sign In
Filter
History
Collections
Today
POST https://packers-backend.herokuapp.com/users/signin
May 7
POST https://packers-backend.herokuapp.com/users/signin
GET https://packers-backend.herokuapp.com/notification
PATCH https://packers-backend.herokuapp.com/form/58f29042
PATCH https://packers-backend.herokuapp.com/form
POST https://packers-backend.herokuapp.com/form
POST https://packers-backend.herokuapp.com/form
POST https://packers-ba
https://packers-b
https://packers-backen
https://packers-backen
localhost/led_data/58e
https://thinktank-backe
https://packers-backen
No Environment
Params
Send
Save
Body
Cookies
Headers (8)
Tests
Pretty
Raw
Preview
JSON

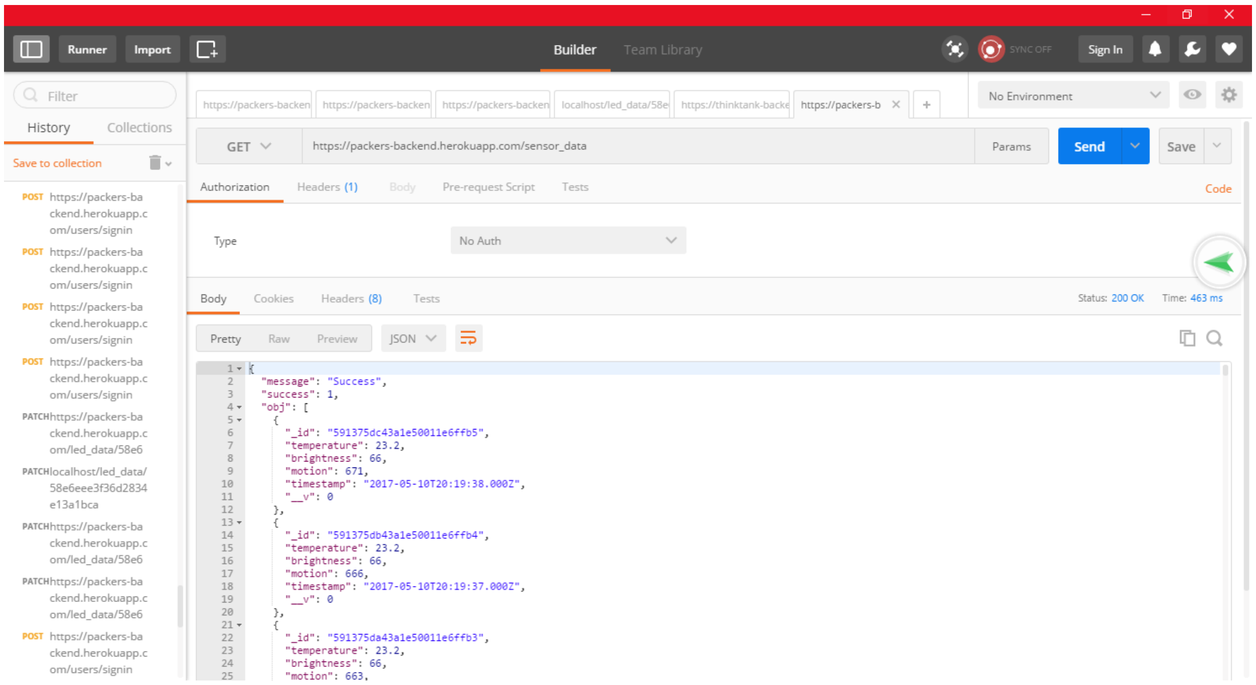
POST https://packers-backend.herokuapp.com/users/signin

```
    "password": "123456",
    "email": "mohamadkhan1993@gmail.com"

}
```

Status: 200 OK Time: 668 ms

```
{
  "message": "Successfully logged in",
  "success": 1,
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
      .eyJ1c2VyIjp7Il9pZCI6IjU4ZjI5MDQyNDUxMTQ2MDAxMTdkMTJhOSIsImZpcnN0TmFtZSI6Ik1vaGFtYWQiLCJsYXN0TmFtZSI6IktoYW4iLCJwYXNzd29yZCI6IiQyYSQxMCQ3VnRyd1l5N3JhdGtFaE9iT
      EVDaC9leGZrOC9yNmpFL1laZmZ2by9ZZUlheWNvZU1hNkRHaSIsImVtYWlsIjoibW9oYW1hZGtoYW4xOTkzQGdtYWlsLmNvbSIsImFkZHJlc3MiOiJGcmVtb250LCBDQSIsIl9fdiI6NiwiZm9ybV9pZCI6WyI
      1OGYyOWJiY2RiMGNhMzI1Zjg2ZGFkNGYiLCI1OGYyOWM4NDAwODczMjAwMTE5MWY1ZDIiLCI1OGYyOWM5MjAwODczMjAwMTE5MWY1ZDMiLCI1OGYyOWM5NTAwODczMjAwMTE5MWY1ZDQiLCI1OGYyOWNhNjAwO
      DczMjAwMTE5MWY1ZDUiLCI1OGYyOWVkOTAwODczMjAwMTE5MWY1ZDYiXX0sImlhdCI6MTQ5NDQ1MjU5NywiZXhwIjoxNDk0NDUyODk3fQ.KKB22Vgviz6YPEn6cOgWCuOJghWTMDJzT60W27LmQFI",
  "userId": "58f2904245114600117d12a9"
}
```


Runner
Import
Builder
Team Library
Sign In
No Environment
History
Collections
Save to collection
POST https://packers-backend.herokuapp.com/users/signin
POST https://packers-backend.herokuapp.com/users/signin
POST https://packers-backend.herokuapp.com/users/signin
POST https://packers-backend.herokuapp.com/users/signin
PATCH https://packers-backend.herokuapp.com/led_data/58e6
PATCH localhost/led_data/58e6eee3f36d2834e13a1bca
PATCH https://packers-backend.herokuapp.com/led_data/58e6
PATCH https://packers-backend.herokuapp.com/led_data/58e6
POST https://packers-backend.herokuapp.com/users/signin
Authorization
Headers (1)
Body
Pre-request Script
Tests
Code
Params
Send
Save
Cookies
Headers (8)
Pretty
Raw
Preview
JSON

GET https://packers-backend.herokuapp.com/sensor_data

Type: No Auth

Status: 200 OK Time: 463 ms

```
{
  "message": "Success",
  "success": 1,
  "obj": [
    {
      "_id": "591375dc43a1e50011e6ffb5",
      "temperature": 23.2,
      "brightness": 66,
      "motion": 671,
      "timestamp": "2017-05-10T20:19:38.000Z",
      "__v": 0
    },
    {
      "_id": "591375db43a1e50011e6ffb4",
      "temperature": 23.2,
      "brightness": 66,
      "motion": 666,
      "timestamp": "2017-05-10T20:19:37.000Z",
      "__v": 0
    },
    {
      "_id": "591375da43a1e50011e6ffb3",
      "temperature": 23.2,
      "brightness": 66,
      "motion": 663,
```

Figure 14: REST APIs Postman

10. CLIENT SIDE DESIGN

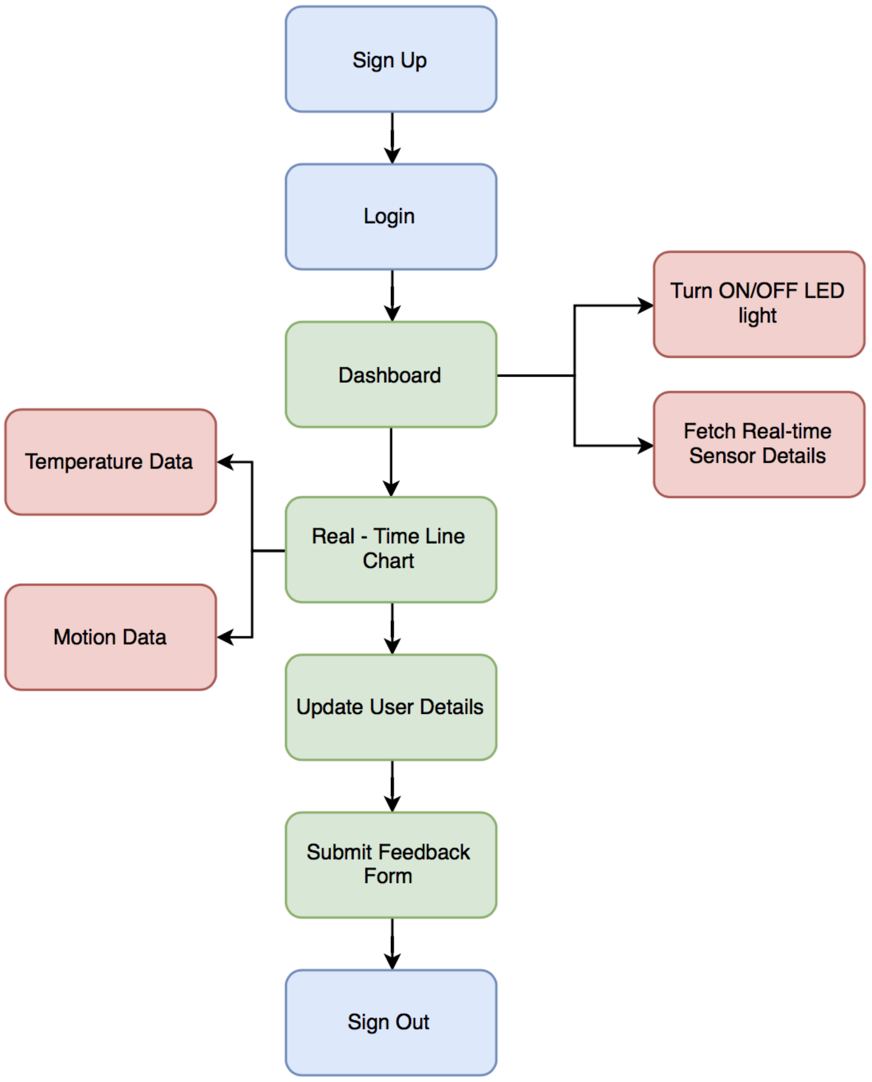
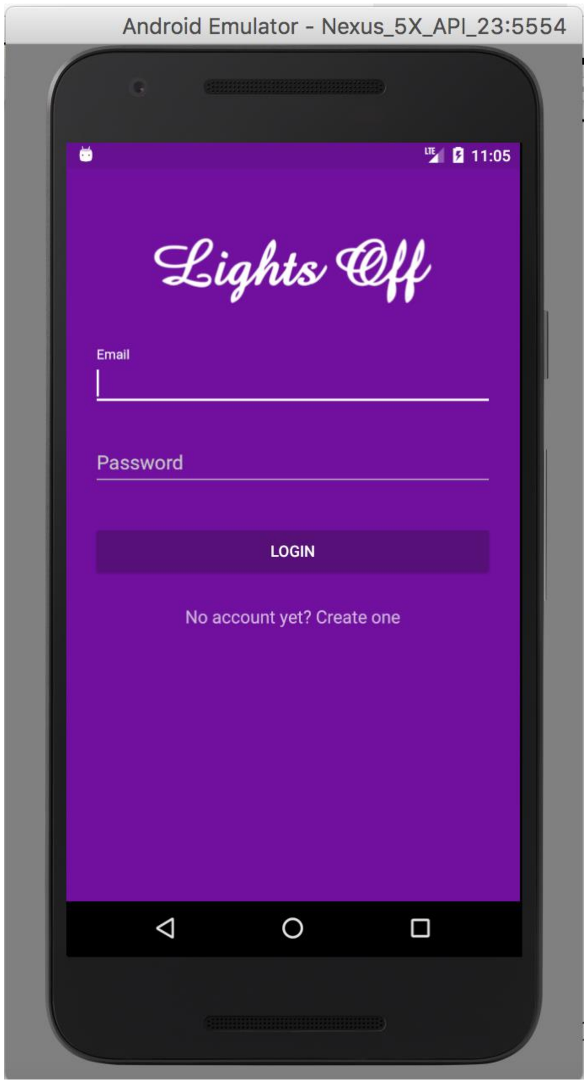
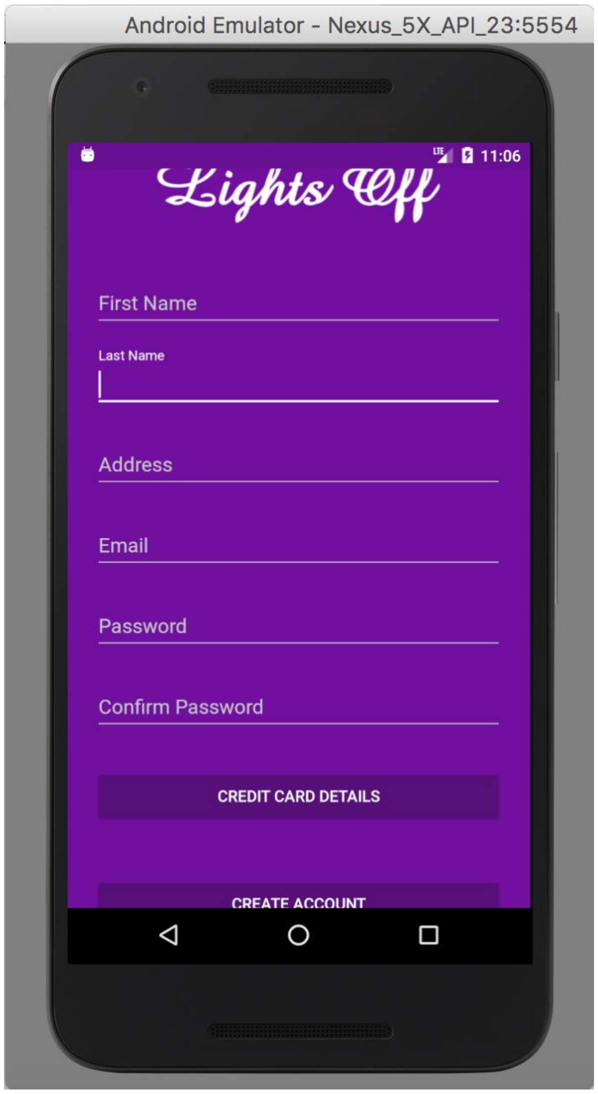


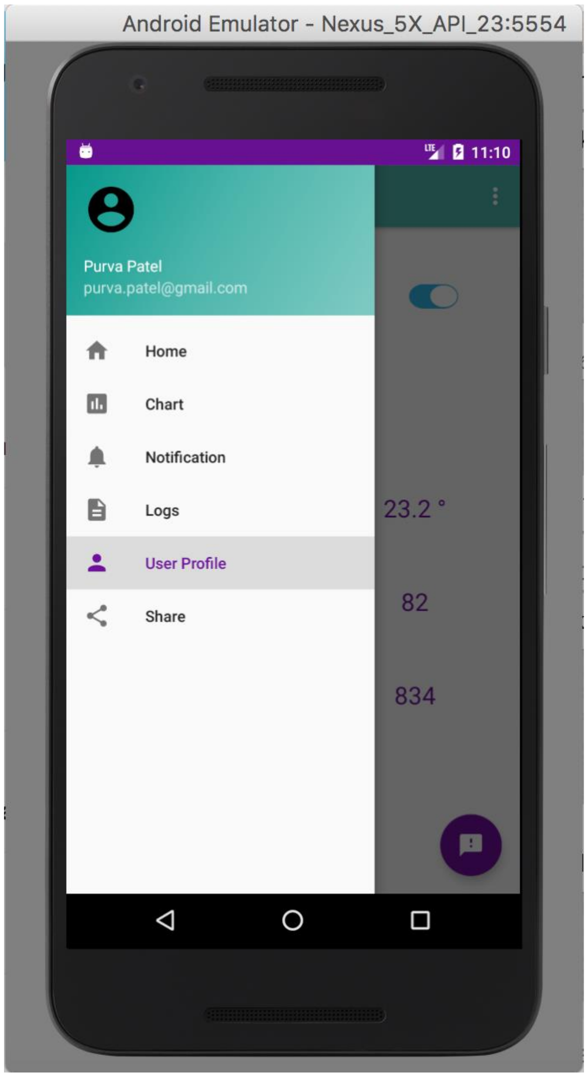
Figure 15: Client Side Design

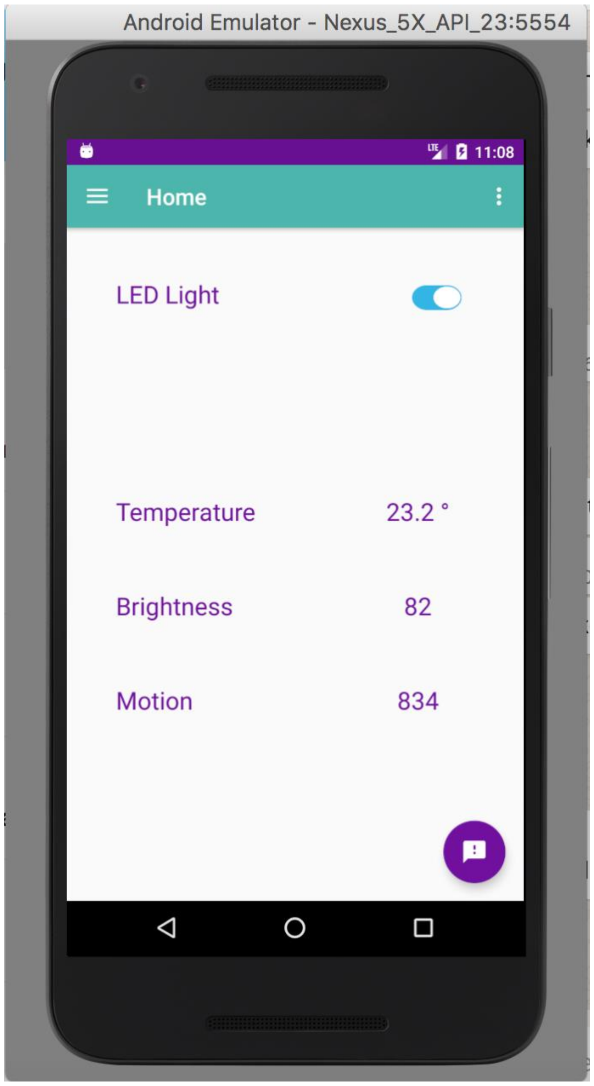
The user initially registers for the application and signs in. On signing in the user can see the dashboard. On the dashboard the user can view the sensor data of Temperature sensor and motion sensor. The user can also turn the LED light ON/OFF. The user can update its details as well. We have also added a feedback form in order for us to improve on the features. Then the user can sign out.

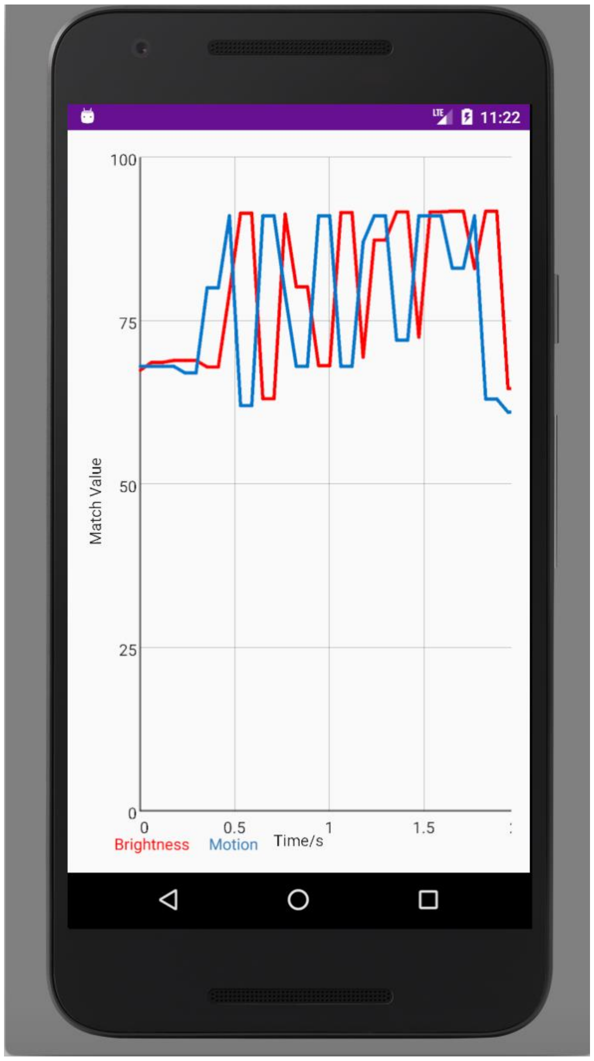
11. UI DESIGN

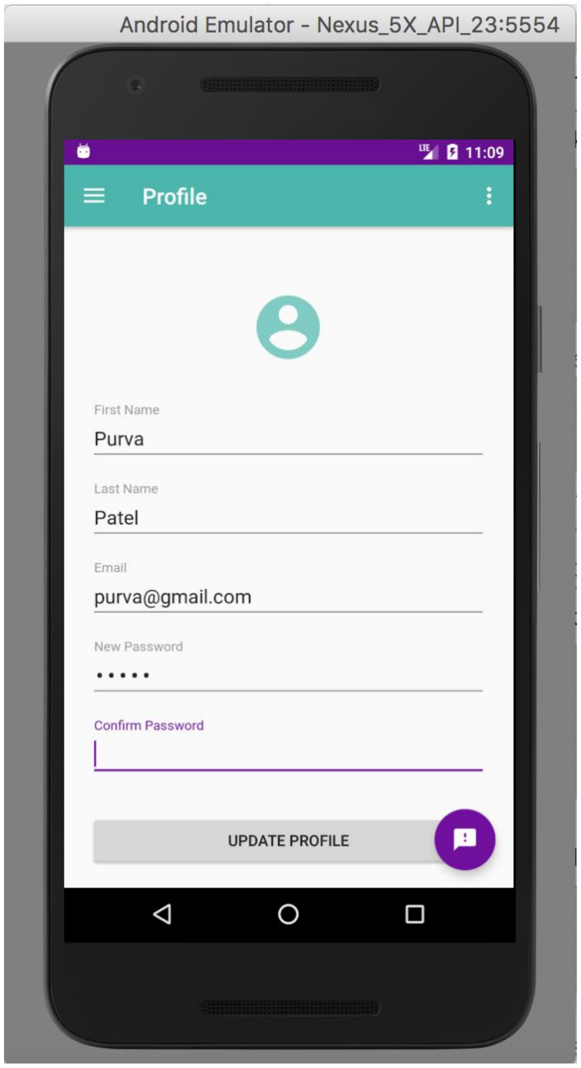

Android Emulator - Nexus_5X_API_23:5554
11:05
Lights Off
Email
Password
LOGIN
No account yet? Create one


Android Emulator - Nexus_5X_API_23:5554
11:06
Lights Off
First Name
Last Name
Address
Email
Password
Confirm Password
CREDIT CARD DETAILS
CREATE ACCOUNT


Android Emulator - Nexus_5X_API_23:5554
11:10
Purva Patel
purva.patel@gmail.com
Home
Chart
Notification
Logs
User Profile
Share
23.2 °
82
834


Android Emulator - Nexus_5X_API_23:5554
11:08
Home
LED Light
Temperature
23.2 °
Brightness
82
Motion
834




Android Emulator - Nexus_5X_API_23:5554

Profile

First Name
Purva

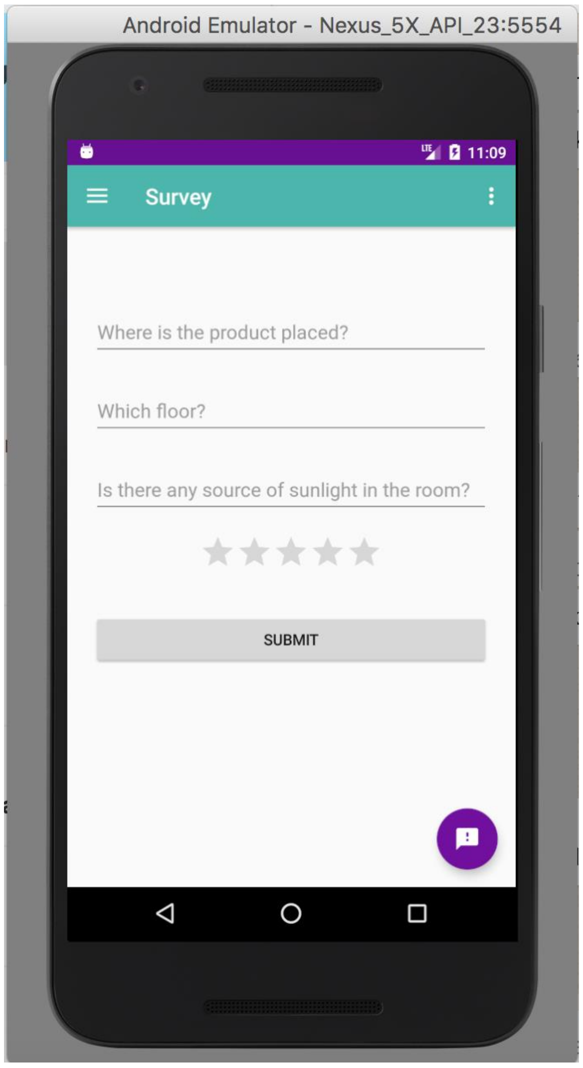
Last Name
Patel

Email
purva@gmail.com

New Password
•••••

Confirm Password

UPDATE PROFILE


Android Emulator - Nexus_5X_API_23:5554
11:09
Survey
Where is the product placed?
Which floor?
Is there any source of sunlight in the room?
SUBMIT

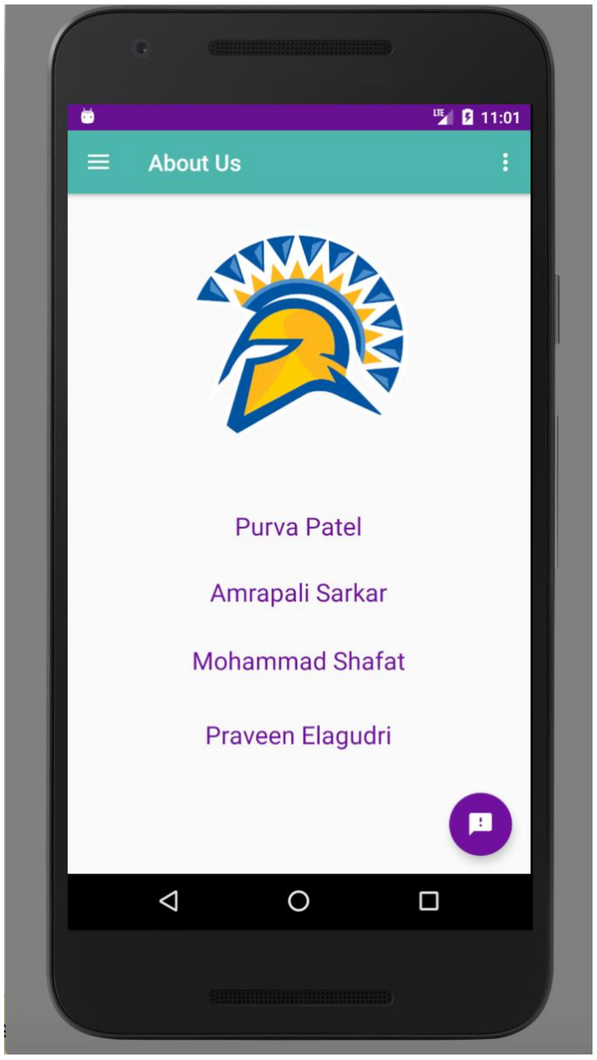


Figure 16: Android App Screen Shots

12. TESTING

Manual Testing: Manual testing is the process of manually testing software for defects. It requires a tester to play the role of an end user whereby they utilise most of the application's features to ensure correct behavior.

Following youtube link are the manual tests performed:

- Youtube link: https://youtu.be/byLJBi4Zkcg

13. AUTOMATION

We have used *Expresso* testing framework for automation testing. Testing user interactions within a single app helps to ensure that users do not encounter unexpected results or have a poor experience when interacting with your app. You should get into the habit of creating user interface (UI) tests if you need to verify that the UI of your app is functioning correctly. A key benefit of using Espresso is that it provides automatic synchronization of test actions with the UI of the app you are testing. Espresso detects when the main thread is idle, so it is able to run your test commands at the appropriate time, improving the reliability of your tests. This capability also relieves you from having to add any timing workarounds, such as Thread.sleep() in your test code.

14. DESIGN PATTERNS USED

14.1 <u>Front End: Model View Controller:</u>

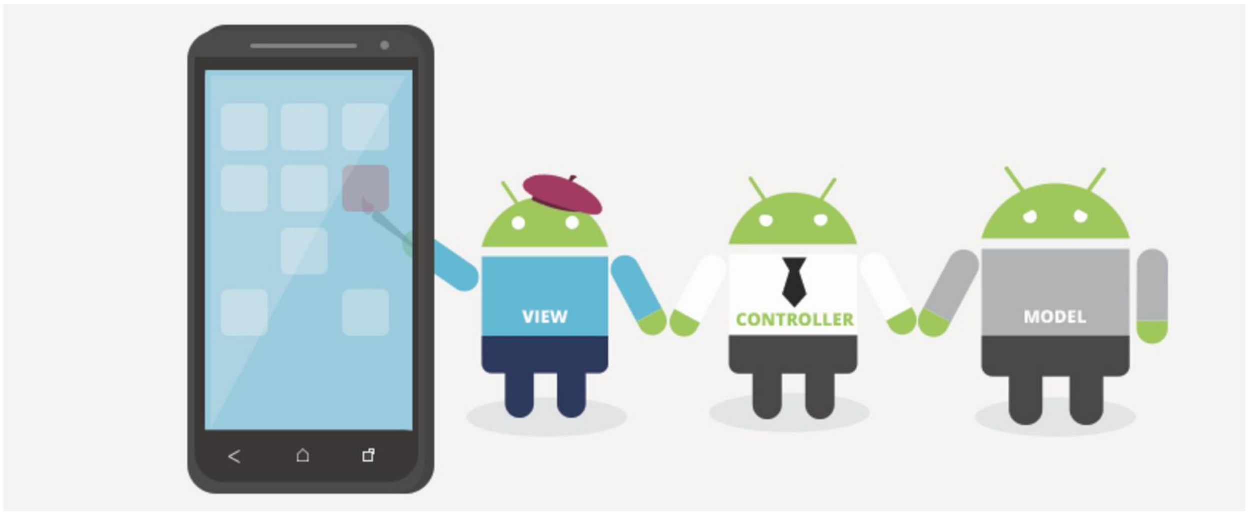


Figure 17: MVC Design Pattern

MVC is already implemented in Android as:

View = layout, resources and built-in classes like Button derived from android.view.View.

Controller = Activity

Model = the classes that implement the application logic

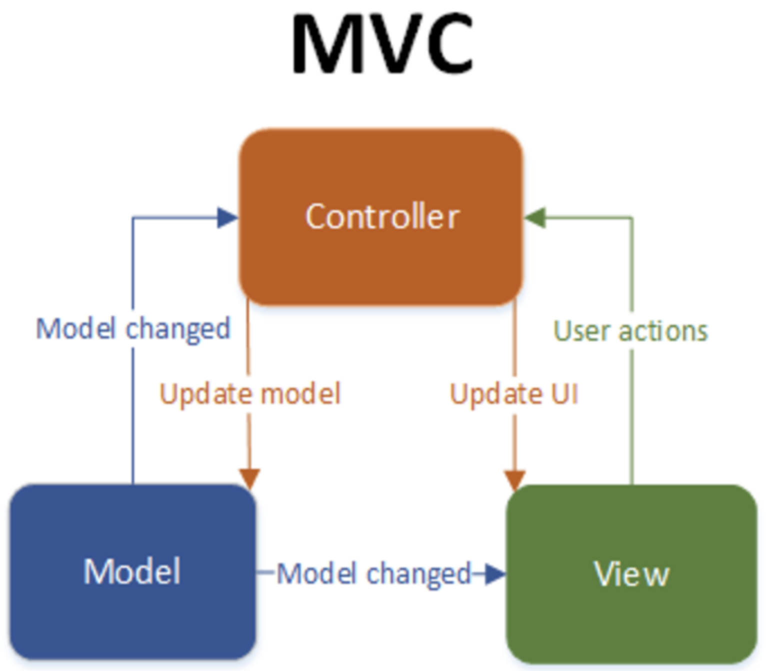


Figure 18: MVC Model

Proper MVP and MVC implementations have the following characteristics:

- Readable and maintainable code
- Modular code which provides high degree of flexibility
- More testable code (especially in context of unit testing)
- Code which is fun to work with

14.2 <u>Data Store: Singleton</u>

The Singleton Pattern specifies that only a single instance of a class should exist with a global point of access. This works well when modeling real-world objects only having one instance. Since it’s accessible from multiple objects, the singleton can undergo unexpected side effects that are difficult to track down – exactly what Future You doesn’t want to deal with. It’s important to understand the pattern, but other design patterns may be safer and easier to maintain.

Middle Tier: Facade

The Facade pattern provides a higher-level interface that makes a set of other interfaces easier to use. The following diagram illustrates this idea in more detail:

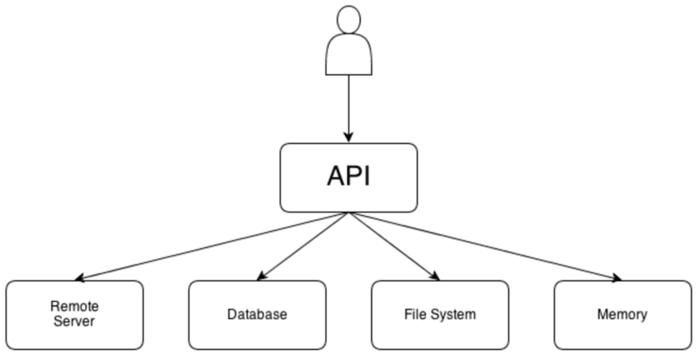


Figure 19: Facade Pattern

Retrofit from SquareOne is an open-source Android library that helps you implement the Facade pattern; you create an interface to provide API data to client classes.

Cloud: Builder

The use of the builder pattern has all the advantages of the first two approaches I mentioned at the beginning and none of their shortcomings. The client code is easier to write and, more importantly, to read. The only critique that I’ve heard about the pattern is the fact that you have to duplicate the class’ attributes on the builder. However, given the fact that the builder class is usually a static member class of the class it builds, they can evolve together fairly easy.

Observer:

Observer pattern is one of the behavioral design pattern. Observer design pattern is useful when you are interested in the state of an object and want to get notified whenever there is any change. In observer pattern, the object that watch on the state of another object are called Observer and the object that is being watched is called Subject. According to GoF, observer pattern intent is define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

Subject contains a list of observers to notify of any change in its state, so it should provide methods using which observers can register and unregister themselves. Subject also contain a method to notify all the observers of any change and either it can send the update while notifying the observer or it can provide another method to get the update.

Observer should have a method to set the object to watch and another method that will be used by Subject to notify them of any updates.

Java provides inbuilt platform for implementing Observer pattern through *java.util.Observable* class and *java.util.Observer* interface. However it’s not widely used because the implementation is really simple and most of the times we don’t want to end up extending a class just for implementing Observer pattern as java doesn’t provide multiple inheritance in classes.

Model-View-Controller (MVC) frameworks also use Observer pattern where Model is the Subject and Views are observers that can register to get notified of any change to the model.

15. LIMITATIONS AND CHALLENGES

Some of the limitations in the proposed system are as follows:

- The Light sensor detect any source of light, for example. Sunlight, Room lights, Bulbs, Screen lights etc. Due to this the system assume that the light in the room is switched on. If the device is kept in a room that may contain sources of light other than the room light, then the system will assume that the light is switched on. One of the common sources of light in a room would be sunlight coming from a window. So, the proposed system does not work accurately during day time.

- PIR motions sensors come with ranges of 10ft, 20ft and 30ft. Thus, any motion occurred outside this range might not be detected.
- If a person in a lighted room is sleeping, meditating, sitting idle or simply is not performing any motion then the system assume that the room is empty.

16. FUTURE SCOPE

To overcome the above limitation, researched can take up the following solutions:

To accurately detect the source of light, machine learning can be implemented in the proposed system. A survey form can be implemented in the android app to get more information about the room like, on which floor the room is? How many sources of light does the room has? Etc.

Higher range motion sensors can be implemented with respect to the size of the room.

Using more sensors like sound sensor in the proposed system can help in accurately finding the empty room.

LIMITATIONS	SOLUTIONS
Daylight	Machine Learning
PIR Motion senses up to 30ft	Use high range sensors
Sleeping, Meditating, Sitting Idle etc. cannot be accurately detected	Use more sensors to detect an empty room accurately like sound sensor etc.

Table 3 : Limitations and Solutions

17. CONCLUSION

In this paper, we propose a new architecture for automated notification, monitoring and control system that uses a Android smartphone and implemented by Arduino UNO microcontroller board and low cost sensors like Temperature sensor, PIR motion sensor, Light sensor and LED sensor. The proposed architecture is used in a quiet based web services in an interoperable application layer for communication between the Android application and the device. All Android-based smartphone, the Wi-Fi connection is the support built, the home access device to control. If the Internet is not possible, it can be access by used the 4G, LTE, 3G or 2G mobile system mobile. Future studies will use the commands for controlling the voice applications by implementing the home server.

18.. PROFILING

NAME	ROLE
Mohamad Shafaat Ali Khan	IoT component
Praveen Elagudri	Backend component
Purva Patel	Android backend
Amrapali Sarkar	UI component and designing

Table 4: Profiling

19. REFERENCES

1. Zaid Abdulzahra Jabbar, R.S. Kawitkar. (2016). Implementation of Smart Home Control by Using Low Cost Arduino & Android Design [Online]. Available:http://www.ijarcce.com/upload/2016/february-16/IJARCCE%2050.pdf
2. Sachin Sonowane, P.B. Borole, Balaji Shrichippa. (2016). Smart Home Monitoring System using ECSN[Online]. Available:https://www.ijircce.com/upload/2016/june/101_Smart.pdf
3. John E. Petersen, Vladislav Shunturov, Kathryn Janda, Gavin Platt and KateWeinberger (2012). Dormitory residents reduce electricity consumption when exposed to real-time visual feedback and incentives[Online]. Available:https://my.vanderbilt.edu/cs265/files/2012/11/Lucid_IJSHE_DormEnergyFeedback1.pdf
4. Prasad Bhukya, Dr. Debasish Basak. (2014). Energy Saving Technologies in Industries-An overview[Online]. Available: http://www.ijsrp.org/research-paper-0414/ijsrp-p2880.pdf
5. Inji Ibrahim Attia, Dr. Hamdy Ashour. (2010). Energy Saving Through Smart Home[Online]. Available: http://www.infomesr.org/attachments/053.pdf
6. Praneet Sah. (2016). Saving Environment Using Internet of Things: Challenges and the Possibilities [Online]. Available:http://file.scirp.org/pdf/AIT_2016091413421916.pdf
7. Anna Florea, Ahmed Farahat, Coruna Postelnicu, Jose L. Martinez Lastra, Francisco J. Azcondo Sanchez (2016). Smart Lighting in Multipurpose Outdoor Environments: Energy Efficient Solution using Network of Cooperating Objects[Online]. Available: http://ceur-ws.org/Vol-1002/paper1.pdf