Mansoura University

Faculty of Computers and Information Systems
Information technology Dept.

SLS: Smart Lab System Using IoT

Graduation Project

by

Ahmed Al-Shahat Al-Hefny

Ahmed Mahmoud Al-Qassas

Aya Gamal Ayad

Aya Mohamed Ibrahim

Doha El-Sayed Rabea

Mahmoud Najah El-Sharawy

Reda Awad Ahmed

Saad Karam Saad El-Sabahy

Salma Ramadan El-Namoury

Youssef Mohamed El-Feky

A project submitted in partial fulfilment of the requirements for the degree of Bachelor of Science Information Technology

Supervised by

Assoc. Prof. Noha Hikal

2018/2019

Abstract

It is obvious that our country, Egypt, is adopting the new technologies to keep up with technological revolution in the world. We believe that modern technologies should be applied in all educational organizations systems, because education is the very beginning step of any success. So, our proposed system will target the educational systems. we are going to build a smart lab system that helps both students and teachers. The system will use IoT technology and mobile application. So, “A Smart Lab System using IoT” is the title of our project and we call the system SLS.

Keywords IoT; Mobile application; E-learning;

Acknowledgement

We would like to thank our great parents who spare no effort for helping and supporting us all the time. And, we promise we will continue doing our best to make them proud of their sons/daughters. We, also, give many thanks to our supervisor, Assoc. Prof. Noha Hikal, who trusted, and helped us and wish her all success in her life. And, for those students who will read this report in the next years, we wish it will help you in your projects. We had a great time in Faculty of Computer and Information Sciences, we learned many things, and now it's the time to use what we'd learnt.

Table of Content

List of Figures

List of Tables

List of Abbreviations

SLS: Smart Lab System

IoT: Internet of Tings

IDE: Integrated Development Environment

Wi-Fi: Wireless Fidelity

RFID: Radio Frequency Identification

IR: Infra-Red

IC: Integrated Circuit

HW: Hardware

PCB: Printed Circuit Board

Chapter 1: Introduction

1.1 Introduction

We are in the age of IoT where electronic devices can connect to a network, make decisions automatically, and provide a user with useful information. Electronic devices in houses, schools, laboratories, markets, and almost every place will be smart to save time and make life much easier. IoT can be considered as a combination of Internet technology and machinery industry to enable machine-to-machine interaction.

In this chapter, we will provide an introduction to our proposed project. First, in section 1.2 and section 1.3 we discuss the motivation and the objectives of our project respectively. Next, in section 1.4, we present the scope of the intended project. Then, Section 1.5 shows timeline of our project. Finally, section 1.6 provides a document organization including a brief description about the context of each chapter.

1.2 Problem Definition

In educational organizations, the time students spend in a classroom is limited. There is a need to spend this time efficiently. Traditional routines waste a lot of time that could have been saved with the help of new technologies. Checking student attendance, quizzing, turning lights on/off, and other similar activities consume a lot of time. However, these activities are too important to be ignored. We seek to use nowadays technology to improve the way these activities are done, make them consume less time, and help the students/teachers focus in what is more important. The idea is so promising. Giving up the traditional methods and adapting new methods/technologies will have great effects on the educational process.

1.3 Project Objectives

We are going to build a system that minimize the time needed for routine activities in a classroom. The following is a list of our project objectives:

- To adjust the lights in a classroom automatically or manually.
- To digitize the process of assigning and evaluating quizzes.
- To allow students to submit their questions or feedback using smartphones.
- To check student attendance automatically.

1.4 Project Scope

The intended system lies under the approach of Internet of Things (IoT). It will apply IoT technology on the educational filed. The use of the system depends on both hardware devices and the use of software means (mobile application). It can be used in classrooms, labs, or auditoriums of educational organizations like schools or universities. The system will digitize many classroom activities instead of using traditional paper sheets or documents.

1.5 Project Timeline

#	Task Name	Start	Finish
1	Background research	20-10-18	31-10-18
2	Determine objectives	01-11-18	20-11-18
3	Review related work and papers	21-11-18	10-12-18
4	Determine needed components	11-12-18	22-12-18
5	Design diagrams	26-01-19	09-02-19
6	Interface design for the mobile application	26-01-19	09-02-19
7	Presentation for phase 1	26-01-19	09-02-19
8	Implementation	10-02-19	15-5-19
8.1	Front-end development	10-02-19	01-03-19
8.2	Gathering hardware components	10-02-19	01-03-19
8.3	Back-end development	02-03-19	04-04-19
8.3.1	Database design	14-04-19	15-05-19
8.3.2	Application functions	14-04-19	15-05-19
9	Testing	15-06-19	26-06-19
9.1	Unit Testing for the hardware part	15-06-19	20-06-19
9.2	Unit Testing for the mobile application part	15-06-19	20-06-19
9.3	Integrating hardware and mobile application	21-06-19	22-06-19
9.4	Integration Testing for the system	23-06-19	26-06-19

Table 1.1: Project timeline

1.6 Document Organization

This document consists of six chapters in addition to one appendix. A brief description about the contents of each chapter is given in the following paragraphs:

Chapter 1, Introduction, introduces the project objectives, the motivation of the project, the approach used in this project, and the scope of the project.

Chapter 2, Literature Review, provides the reader with an overview of the previous related work, common technologies used, and the relation between our work and the relevant work.

Chapter 3, System Analysis, includes the analysis of existing system, system requirements, use requirements, system architecture, development methodology using UML, the tools, and languages used in our system.

Chapter 4, System Design, provides the system design including class diagram, database design and interface design.

Chapter 5, System Implementation, shows the process of mapping design into implementation, sample application code, system testing, results of the investigation, and goals achieved.

Chapter 6, Conclusion and Future Work, summarizes the entire research, and addresses the suggested improvements for the system.

Appendix A contains information about the contents of the project DVD and how to use them.

Chapter 2: Literature Review

2.1 Introduction

In this chapter, we provide a background about the tools and techniques needed to build our system. We also review the previous related work to our system, and the common technologies that are used. In the end of the chapter, we provide a comparison between the system we are going to build and the related systems.

2.2 Background

The system we are going to build, SLS, will depend on hardware components, and a mobile application to operate. The following subsections provide a brief background information about the equipment, tools, and techniques needed to build our system including microcontrollers, Arduino boards, Arduino IDE, simulators, Relays, Ethernet modules, Wi-Fi modules, RFID (Radio-frequency identification), IR sensors, programming languages, and libraries.

2.2.1 Microcontrollers

The improvement of integrated circuits has led to the appearance of a new generation of electronic circuits called microcontroller. A microcontroller acts as a small computer that can be programmed to achieve many jobs such as reading temperature or controlling electrical motors. A typical microcontroller includes a processor, memory and input/output peripherals on a single chip. The key difference between microcontroller and microprocessor is that in microcontrollers, RAM, ROM, EEPROM are embedded while they are external in case of microprocessors.

2.2.2 Arduino

Arduino, which is a very common microcontroller, is an open source electronics platform that depends on the use of hardware and software together. It’s used to read an input from a peripheral device (i.e. a sensor) then turn it into an output depending on the set of instructions given to the Arduino board. [1] Arduino was born at the Ivrea Interaction Design Institute as an easy tool to help students without a background in electronics and programming. Later, it started changing to adapt new challenges. Arduino users can adapt the boards to their needs, update them, or distribute their own versions. It has many advantages. It’s cheap, easy to use, simple to program, and can be composed with projects that are developed using programming languages such as MATLAB, JAVA, and VB.NET. The list of Arduino boards Includes Arduino Uno, Arduino Due, Arduino Mega, Arduino Leonardo, and others. These types differ from each other in the processor, memory, and used number of I/O pins. Arduino Uno, which is the most popular, has 14 digital I/O pins (6 of them can be used as Pulse Width Modulation output) and 6 analog input pins. *Figure 2.1* shows an illustration of the Arduino UNO board. *Table 2.1* shows the usage of the main components of the Arduino Uno board.

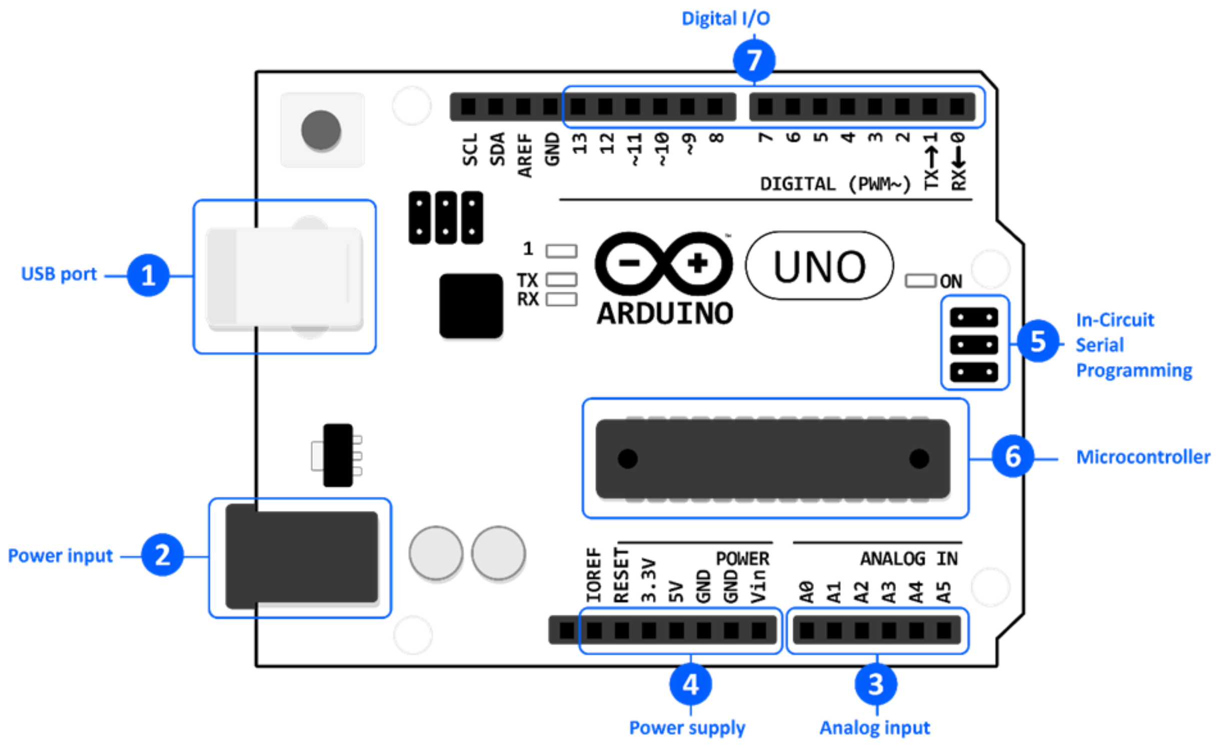


Figure 2.1: Illustration for the Arduino UNO board components.

2.2.1 Programming language and Arduino IDE

An Arduino board is programmed using the Arduino programming language and the Arduino IDE. The programming language of the Arduino is a simple form of C and C++. It is based on what is called “sketches”. A sketch is the code that is uploaded to be run on the board. The simplest Arduino sketch consists of only two functions which are the **`setup()`** function and the **`loop()`** function. The former runs only one time when the sketch begins for initialization purposes, while the latter is used for repeating the code to be executed as long as the device is working.

#	Component	Description
1	**USB port**	Allows you to transfer your sketch into the Flash memory of the microcontroller. It can be used also to supply power to the Arduino.
2	**Power input**	Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack.
3	**Analog input**	These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value
4	**Power supply**	• 3.3V pin − Supply 3.3 output volt • 5V pin − Supply 5 output volt • GND pin − There are several GND pins on the Arduino, any of which can be used to ground your circuit. • Vin pin − This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.
5	**In-circuit serial programming (ICSP)**	Allows the Arduino to be programmed while installed in a complete system, rather than requiring the chip to be programmed prior to installing it into the system.
6	**Digital I/O**	These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc.
7	**Microcontroller**	You can assume it as the brain of your board. The main IC (Integrated Circuit) on the Arduino is slightly different from board to board.

Table 2.1: The usage of the main components of the Arduino Uno board.

2.2.2 Arduino Simulator

Arduino simulators make it possible for circuit designers to learn, program and test ideas before implementing them without worrying about wasting time and money. One great simulator is Proteus. Operating on a PC, Proteus can simulate the interaction between electronic devices connected to the microcontroller and the software programs running on it.

2.2.3 Relay Module

Circuits that operate at high voltages or high currents cannot be controlled directly by an Arduino. So, we need to isolate the Arduino form the controlled circuit. A relay is a micromechanical switch that uses an electromagnet to mechanically operate a switch. The switch, running on a small current, turns a larger current ON or OFF. We can control a relay operation (ON/OFF) by the code uploaded on the Arduino board. Relays are, also, necessary when multiple circuits need to be controlled using only a single signal.

2.2.4 Ethernet Module and Wi-Fi Module

Our system requires the Arduino to be connected to the internet. To connect the Arduino to the internet, we can use Ethernet module for wired internet connection or Wi-Fi module for wireless connection. *Figure 2.2 (a)*, and *Figure 2.2 (b)* show the Ethernet and Wi-Fi modules respectively. In our proposed project, we will use ethernet module. Later in chapter 5, we will provide a description of how to connect the Arduino board and the ethernet module.

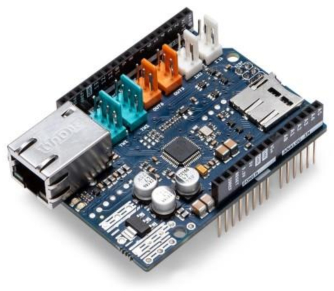


Figure 2.2 (a): Ethernet module [2]

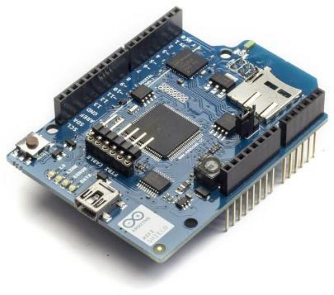


Figure 2.2 (b): Wi-Fi module [3]

2.2.5 RFID (Radio-Frequency Identification)

Radio-frequency identification is a technology based on the use of radio waves to identify the information that is stored in a tag. One advantage is that the tag doesn't need to be in the reader's direct line to be tracked. The system of RFID is made of two parts: a tag and a reader. Information is stored in the tag which is attached to some object. A tag can be active, or a battery-assisted passive. Active tags are powered by a battery and periodically sends an ID signal. Passive tags depend the electromagnetic field that affects a circuit inside it. *Figure 2.2* illustrates how RFID technology works. The Tag is usually a passive component, which consist of just an antenna and an electronic microchip, so when it gets near the electromagnetic field, a voltage is generated in its antenna and this voltage serves as power for the microchip. The RFID module will be connected to the Arduino board to provide the input data to our system.

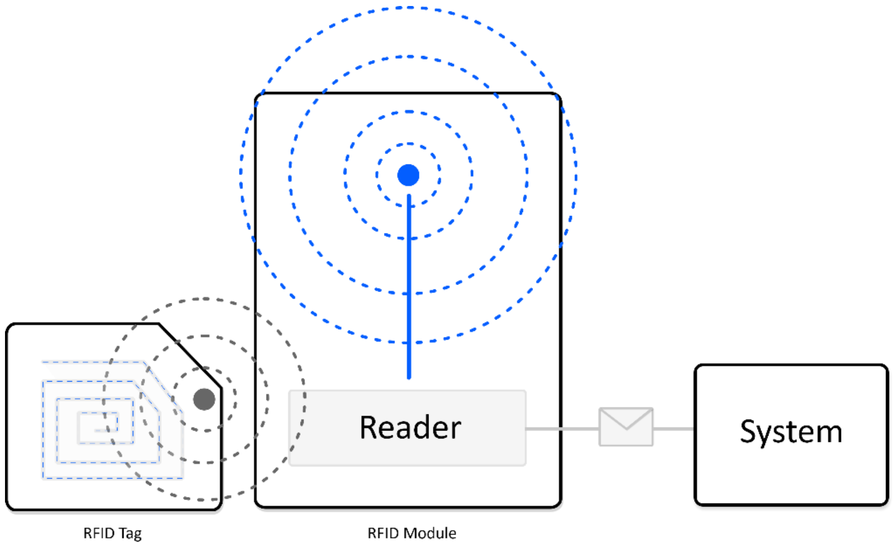


Figure 2.3: How RFID technology works.

2.2.6 IR (Infra-Red) Sensors

One of the most popular sensors is IR (Infra-Red) sensor. An IR sensor consists of an IR LED and a photo diode. The way it works is that the IR LED emits IR radiation then photo diode sense that IR radiation. The IR LED and photo diode can be placed directly where IR radiation falls on photo diode which sense it and that means there is no objects between them. It can, also, be placed indirectly. When an object place in front of IR Pair, The IR light reflected and absorbed by photo diode. *Figure 2.1(a)*, and *Figure 2.1(b)* illustrate the difference between the two ways. There are two types of IR sensor. Active IR sensors employ both infrared source and infrared detectors. Passive IR sensors are just IR detectors as they don't use any IR source. *Table 2.2* shows a comparison between RFID technology, which we are going to use in our system, and IR sensors which are used in many related systems.

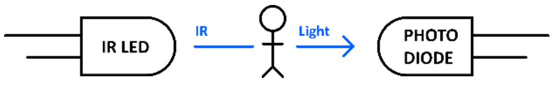


Figure 2.4 (a): Placing the IR pair directly.

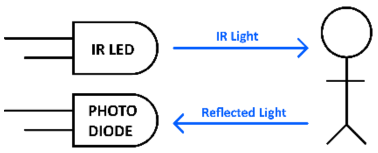


Figure 2.4 (b): Placing the IR pair indirectly.

	RFID	**IR**
Working principle	Electromagnetic waves propagated by an antenna	Detecting Infra-Red light
Range	Short range	Short and medium range
Advantages	• Tags are simple to installed/embedded • Tags can store data up to 2KB • Cannot be easily replicated which increases the security • Tags don't need to be in line of sight	• Low energy requirements • No radiation rays
Disadvantages	• Active tags can be very expensive	• Support only line-of-sight access. It doesn't pass through opaque or solid obstacles.
Application example	People/Asset tracking	Motion detection

Table 2.2: Comparison between RFID and IR.

2.2.7 Native and Cross Platforms

Before developing a mobile application, you should determine whether it will be designed for a particular operating system, or multiple operating systems. A native mobile application is an application that is made for a particular operating system (i.e. Android). That, a native Android mobile app will not operate on an iOS device (i.e. iPhone) and vice versa. The use of cross-platform makes it easy and less expensive to develop mobile apps that are compatible with multiple operating systems. Both mentioned techniques have their advantages and disadvantages. The choice between them depends on the intended scope of application. Our intended mobile application is supposed to provide some interaction between the user and the Arduino. The user of the application might have an Android or iOS device. So, using a cross-platform would be our choice.

2.2.8 REACT Native

REACT Native is a framework based on JavaScript. It is used to write mobile applications works on both iOS and Android devices. It’s based on JavaScript library, made by Facebook, for building user interfaces (REACT). Instead of dealing with browsers and web views, it deals with mobile platforms. To use React native framework, you should be familiar with REACT JS and basic web technologies like HTML, CSS, CSS3, JavaScript and OOP. The application developed by REACT Native will render using real mobile UI components not web views. Also, your apps can access platform features like the phone camera, or the user’s location.

2.3 Review of Relevant Work

The following are some of researches and papers that studies the use of IoT technologies in the education field:

1. Maryam Bagheri and Siavosh H. Movahed performs a research on “The Effect of the Internet of Things (IoT) on Education Business Model” [5]. They had categorized the application of IoT in education into four groups: energy management and real time monitoring, monitoring student’s healthcare, classroom access control, and improving teaching and learning. Our project works in the last two categories which are classroom access control and improving learning process.

2. Prof. Rohini Temkar, Mohanish Gupte, et al. made the paper “Internet of Things for Smart Classrooms” [6]. The paper describes how IoT can be implemented for a better

classroom experience. And shows some scenarios of practical processes that may happen in a classroom.

3. M.R.M.Veeramanickam and Dr. M. Mohanapriya made the paper "IOT enabled Futurus Smart Campus with effective E-Learning : i-Campus" [7]. They analyzed the efficacy and need of Internet of things in Smart College Campus. And discusses the promising future of IoT technologies in E-Learning field.

Considering the control of lights, fans, projectors, or any of electrical devices that could be in a lab/lecture's hall using of a microcontroller with/without Android application, the following is some related work:

1. Ying-Wen Bai and Yi-Te Ku have designed a "Automatic Room Light Intensity Detection and Control Using a Microprocessor and Light Sensors" [8]. Their design, the HLCM (Home Light Control Module) detects if a human body enters the detection area or not to turn on/off lights. If there is, the HLCM maintains sufficient light by controlling the number of light LEDs. They use a PIR (Passive Infra-Red) sensor circuit to detect whether someone is passing through the detection area or not. The PIR sensor has a wired connection to a microprocessor. Depending on the number of people detected by the PIR sensor, the system turns ON/OFF light LEDs automatically. Their purposed HLCM design does not provide manual control over the system using smartphones.
2. J.Chandramohan, R.Nagarajan, et al. have made the design of "Intelligent Smart Home Automation and Security System Using Arduino and Wi-fi" [9]. They use the Arduino UNO as a microcontroller. Their design involves the use of an Android application for controlling the system by a smartphone. The connection between the mobile application and the Arduino is maintained by a Wi-Fi module.
3. Archana D, Rajani B.R, et al. have designed a "Bidirectional Visitor Counter for Smart Power Management" [10]. Depending on Arduino UNO and IR sensor, their system calculates the number of people in the hall to control the lights and fans considering the environmental aspects like temperature, light, etc. If no one in the room, the fans and lights are going to switched off automatically. The system operates automatically, and no mobile app is used for manual control on the system.
4. Gaurav Waradkar, Hitesh Ramina, et al. have designed a "Automated Room Light Controller with Visitor Counter" [11]. The system is used to control Room Lights (LEDs) as well to count the number of persons entering the room automatically using

IR sensors and a controller. No mobile application was involved, and the system operates automatically.

5. Nabeel Salih Ali, Ali Al Farawn, et al. have designed “Attendance and Information System using RFID and Web-Based Application for Academic Sector” [12]. The system aims at supporting the attendance management system for an academic sector in the usage of the RFID technology and microcontroller board.

2.4 Relationship between the Relevant Work and Our Work

The previously mentioned related work shows that designed systems are either automated systems or controlled manually using smartphones. We seek to design a system with the two options available. That, you can control lights (i.e. lights) within the classroom manually or automatically. *Table* 2.3, and *Table 2.4* show the comparison between our systems and other relevant systems in terms of objectives, and components respectively

	Home Light Control Module	Intelligent Smart Home Automation and Security System	Bidirectional Visitor Counter for Smart Power Management	Automated Room Light Controller with Visitor Counter	Attendance and Information System using RFID	Smart Lab System (SLS)
Manual light control	✗	✓	✗	✗	✗	✓
Automatic light control	✓	✗	✓	✓	✗	✓
Automatic Check attendance	✗	✗	✗	✗	✓	✓

Table 2.3: A comparison between objectives of our system and other systems.

	Home Light Control Module	Intelligent Smart Home Automation and Security System	Bidirectional Visitor Counter for Smart Power Management	Automated Room Light Controller with Visitor Counter	Attendance and Information System using RFID	Smart Lab System (SLS)
Mobile/Web Application	✗	✓	✗	✗	✗	✓
RFID tags	✗	✗	✗	✗	✓	✓
IR sensors	✓	✗	✓	✓	✗	✗
Ethernet module	✗	✗	✗	✗	✗	✓
Wi-Fi module	✗	✓	✗	✗	✗	✗
Use server?	✗	✗	✗	✗	✓	✓

Table 2.4: A comparison between components of our system and other systems.

2.5 Summary

Internet of Things is a very vital field in our time. Sooner or later, all electronic devices are to be smart. The pervious sections show the needed background information to understand how we are going to add smartness to an ordinary lab. Section 2.1 provides information about hardware devices, software tools, and techniques needed to develop the Arduino, the mobile application that is compatible with multiple operating system, and the interaction between the Arduino and the mobile application. Section 2.2 discusses relevant work based on the use of sensors and microcontrollers to count the number of people within an auditorium and then make some action. Some of projects intended to make a system that operates automatically, others depend on manual control of the user using a mobile device. In section 2.3, we discuss the relationship between our intended project and the related work. We seek to build a system that provides both automatic and manual control. And, also, provides some useful features related to activities could be done within a lab.

Chapter 3: System Analysis

3.1 Introduction

In this chapter we provide a detailed knowledge about our system including functional requirements, non-functional requirements, user requirements, system architecture, use case, and sequence diagrams. We can divide the process of analysis into three parts, which are determining requirements, determining system architecture, and determining development methodology. *Section 3.2*, *Section 3.3*, and *Section 3.4* will discuss those three parts. In the end of this chapter, we provide information about the tools and languages we needed to use to build our system.

3.2 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.

3.2.1 Functional Requirements

In systems engineering, a functional requirement defines a function of a system or its components, 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 3.1* shows the functional requirements for our Smart Lab System.

3.2.2 Non-functional Requirements

The system needs both hardware components and software components to operate efficiently and meet the requirements. Any failure of the components of the systems may lead to one or more of functional 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 3.2* shows the non-functional requirements for the system to operate efficiently.

#	Functional requirement	Description
1	Monitor student attendance	The system should determine the attendant students in the lab and it should provide names of attendants through a mobile application.
2	Control light system automatically	The system should adjust the lights in the lab automatically according to attendants automatically.
3	Allow manual control to light system	The system should allow the teacher only to turn off automatic light control and adjust lights manually.
4	Permit the teacher to assign quizzes	The system should allow the teacher only to create quizzes and submit them.
5	Display each submitted quiz to the attendants	The submitted quizzes should be accessible only to the attendants in the lab to answer it.
6	Evaluate the quizzes and display student degrees	For each student submitted his/her answers. The system should evaluate these answers and display the degrees to the teacher through the application.
7	Allow attendants to post questions	The system should allow each attendant student to post one or more question through the application.
8	Allow students/teacher to view question	The system should allow the teacher and the students to view the posted questions through the application.
9	Prevent students outside the lab from logging in	The system should not allow the students who were not identified to be attendants to login.

Table 3.1: functional requirements for Smart Lab System

#	Non-Functional requirement	Description
1	Minimize response time	Response time should be 1~2 seconds for the identification method placed on the door of the lab.
2	Well-designed light system	Light bulbs in the lab should be well distributed according to places where attendants sit.
3	Ease-of-access	The mobile application design should be very simple, clear, and easy to use.
4	Minimize efforts	The mobile application should minimize actions needed to do some function.
5	Portability	The mobile application should be available for Android and iOS operating systems.
7	Availability	The system should be available to all attendants.

Table 3.2: Non-functional requirements for Smart Lab System.

3.2.3 User Requirements

The user of the SLS can be either a student or a teacher. Both have some different specifications from each other. The following two lists shows the requirements for both:

A. Student requirements

1) Identification method (RFID technology)
2) Login to be checked as an attendant
3) Like questions added by other students
4) Answer quizzes
5) Post one or more questions
6) View student questions

B. Teacher requirements

1) Identification method (RFID technology)
2) Control light system
3) Assign quizzes
4) View quizzes results
5) View the number of attendants
6) View student questions

3.3 System Architecture

After determining the requirements of the system, we will describe its major components, their relationships (structures), and how they interact with each other. *Figure 3.1* is an illustration of a classroom with our intended system installed in it. At the door of the lab, there is the identification method (RFID module). When a person shows its RFID tag, the module will identify it. The RFID module will send the data stored on the tag to the Arduino which will do two jobs. First, it will adjust the light system in the lab if needed. Second, The Arduino, which is connected to the internet using Ethernet module, will send the students data to the server. Now, if the student is valid, he'll be able to login, checked as an attendant, use the application functions. The process applies for the teachers but with different functions provided by the application. *Figure 2.3* shows the SLS architecture with the main components and connections between them.

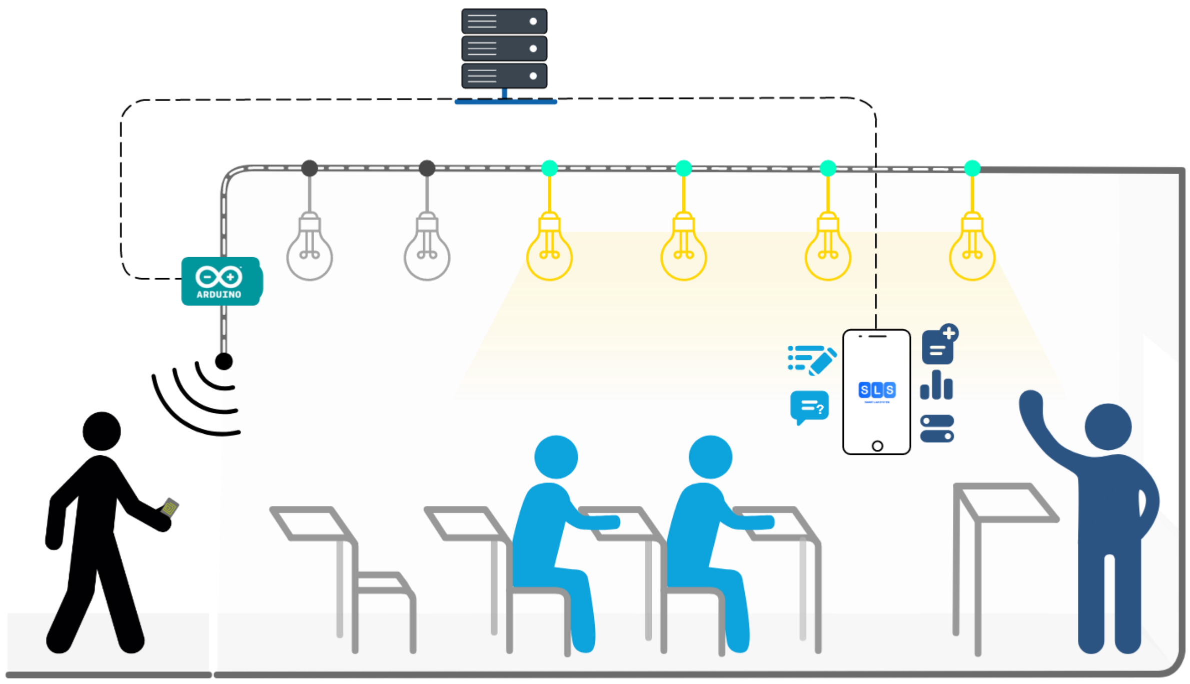


Figure 3.1: An illustration of a classroom with SLS installed in it.

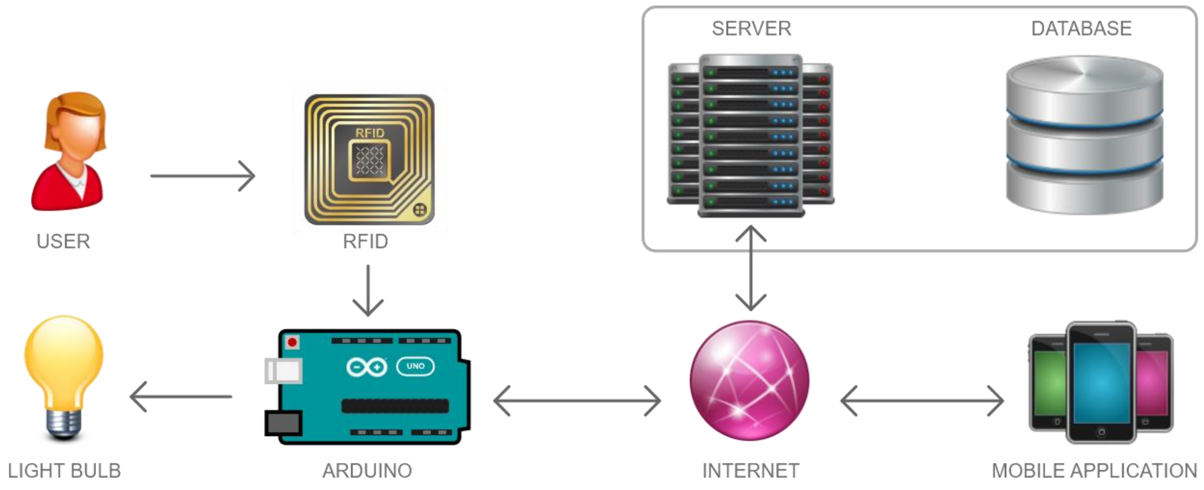


Figure 3.2: SLS Architecture.

3.4 Development Methodology

After we knew the basic structure of the SLS. We are going to view all of its functions, the relation between them, and the sequence of their executions in the following subsections.

3.4.1 Use Case Diagram

First, with use case diagram, we will specify the expected behavior (what) of the system, not the exact method of making it happen (how). This helps us to design the system from end user's perspective. *Figure 3.3* show the use case diagram where the system boundary is the lab and the actors are students and teacher.

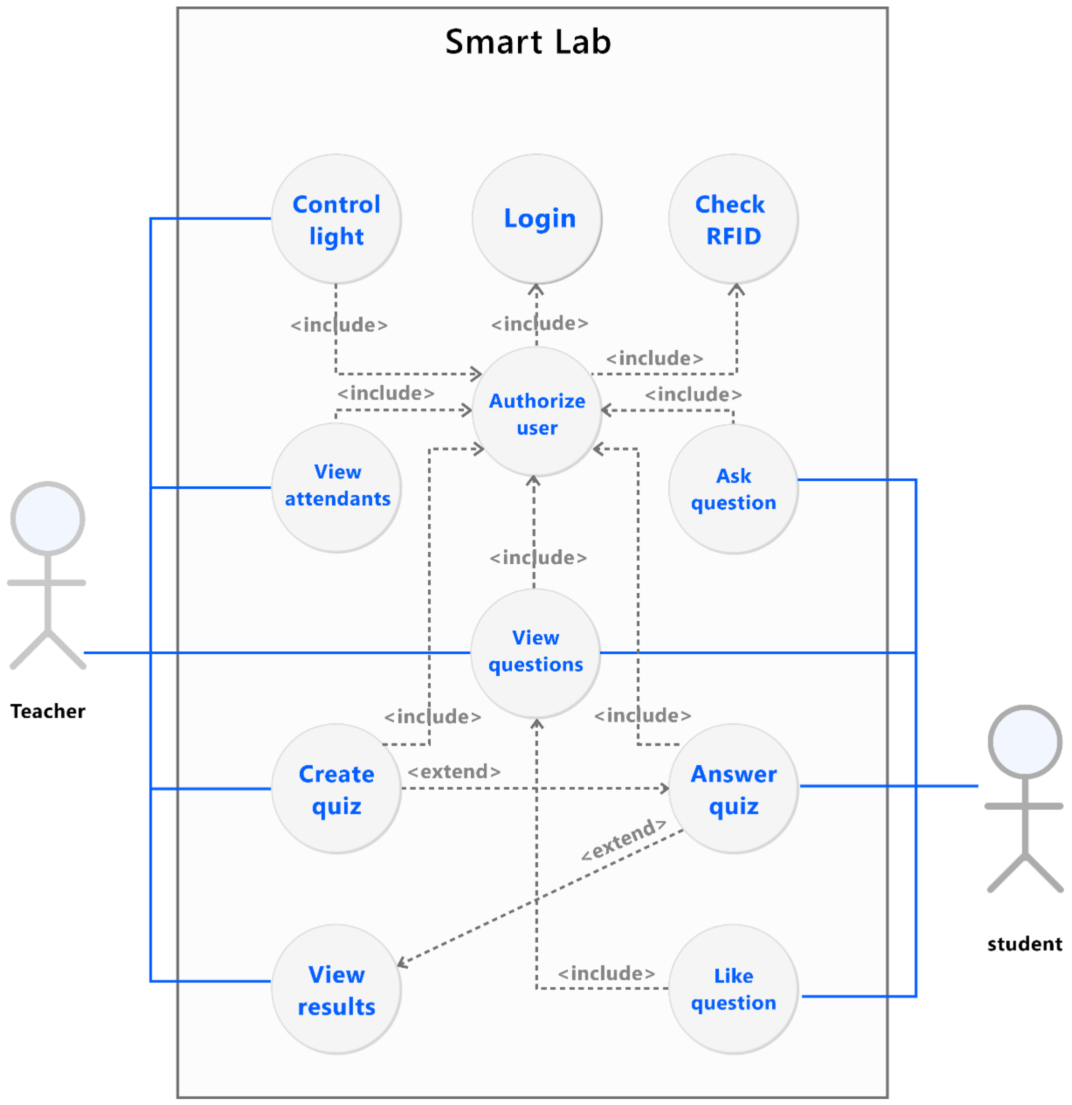


Figure 3.3: Use case diagram.

3.4.2 Sequence Diagrams

Having an idea about the operations available for each user and the relations between them, we are going to dig deeper and see how they are done. The sequence diagrams will show you how an operation is done and the inner details of it. To use our system, the user must be authorized first. The process of authorization includes two phases. The user should prove that he/she is inside the lab. This is done using RFID tags. After the user proves being in the lab, he/she can now login (i.e. using id and password). Once the user logged in successfully, his/her corresponding record in the database is updated to be attendant. In case of failure in any of the two phases, the user will neither be attendant nor use the application. *Figure 3.4* shows a sequence diagram for the authentication process.

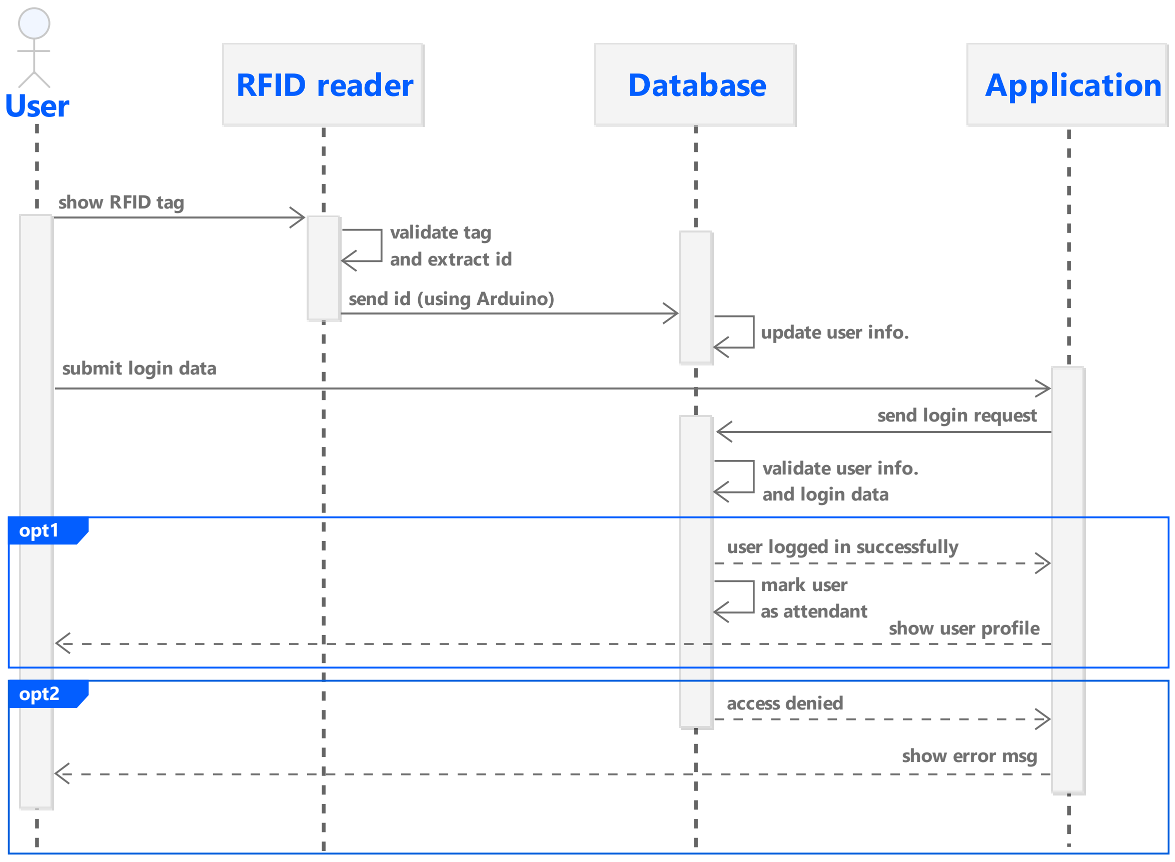
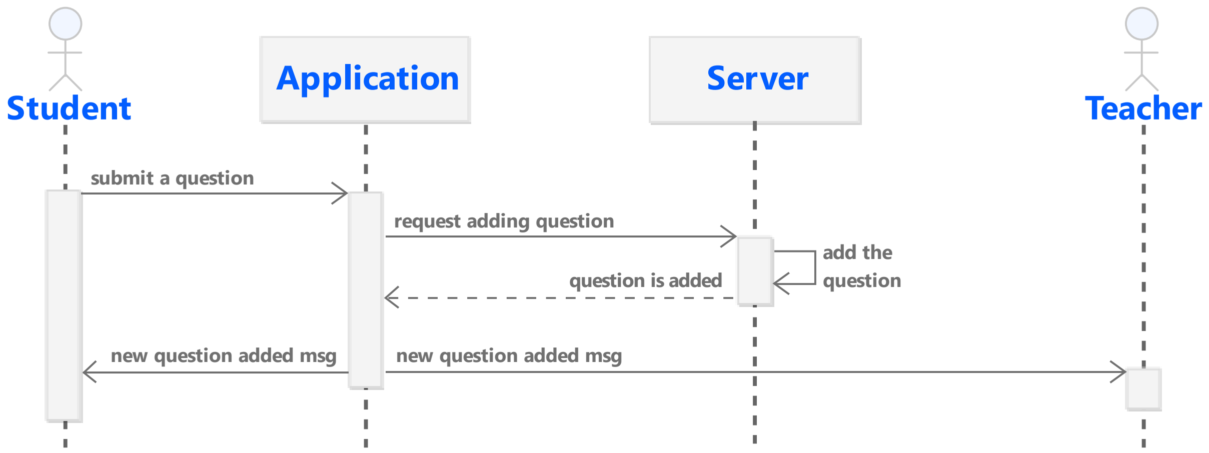


Figure 3.4: A sequence diagram for authorization.

After login, both teacher and students can use the mobile application. If you are a student, you will be able to ask questions. Either you are a student or a teacher, you can view all the questions of the student through the application. *Figure 3.5* shows the sequence diagram for adding a question. Once a new question is added, teacher and students will see



a notification in the application.

Figure 3.5: A sequence diagram for adding and viewing questions.

While a student can add a question through the application, the teacher can create a quiz for the students. To create a quiz, the teacher will add some multiple-choice questions with the correct answer for each. When the teacher submits the quiz, it's sent to server which sends only the questions to the students. Each student can answer the question and submit the answers back to the server. The server evaluates all the students answers and sends the results to the teacher. *Figure 3.6* shows the sequence diagram for the quizzing process.

The remaining operations are controlling light system and checking attendance. These two operations are provided only to the teacher in the lab. The former operation depends on detecting how many persons in the lab, while the latter operation depends on identifying who are the persons in the lab. Basically, both require the use of RFID technology. *Figure 3.7* and *figure 3.8* shows the sequence diagrams for controlling light and checking attendence respectively.

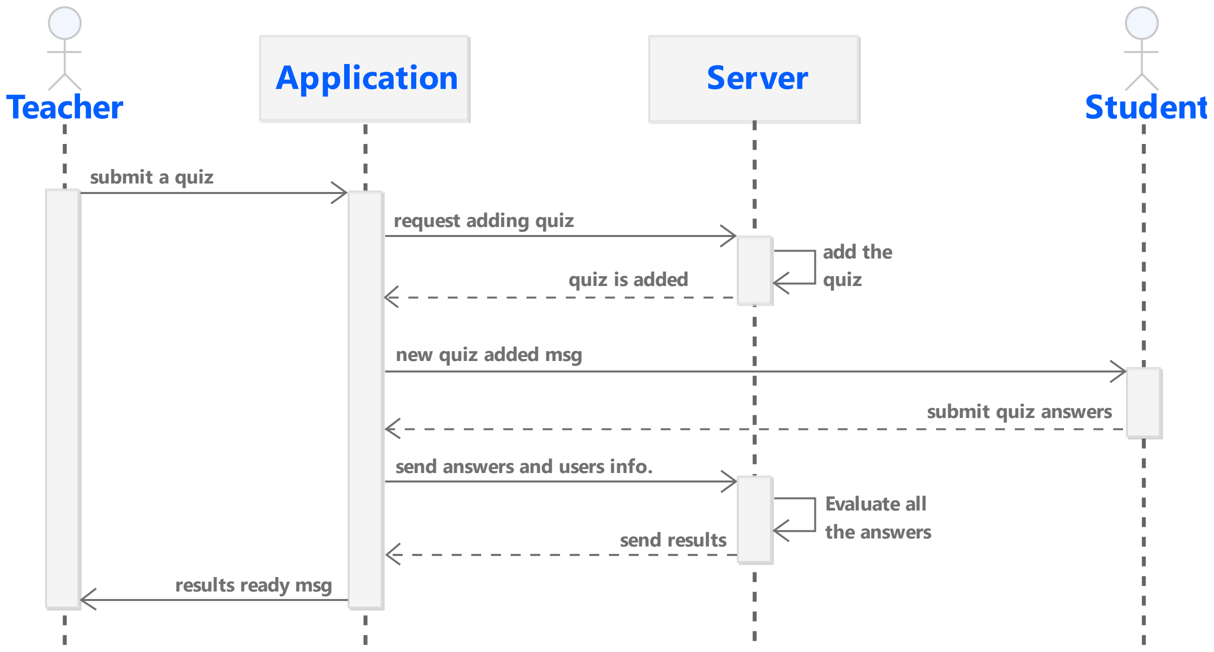


Figure 3.6: A sequence diagram for adding and evaluating a quiz.

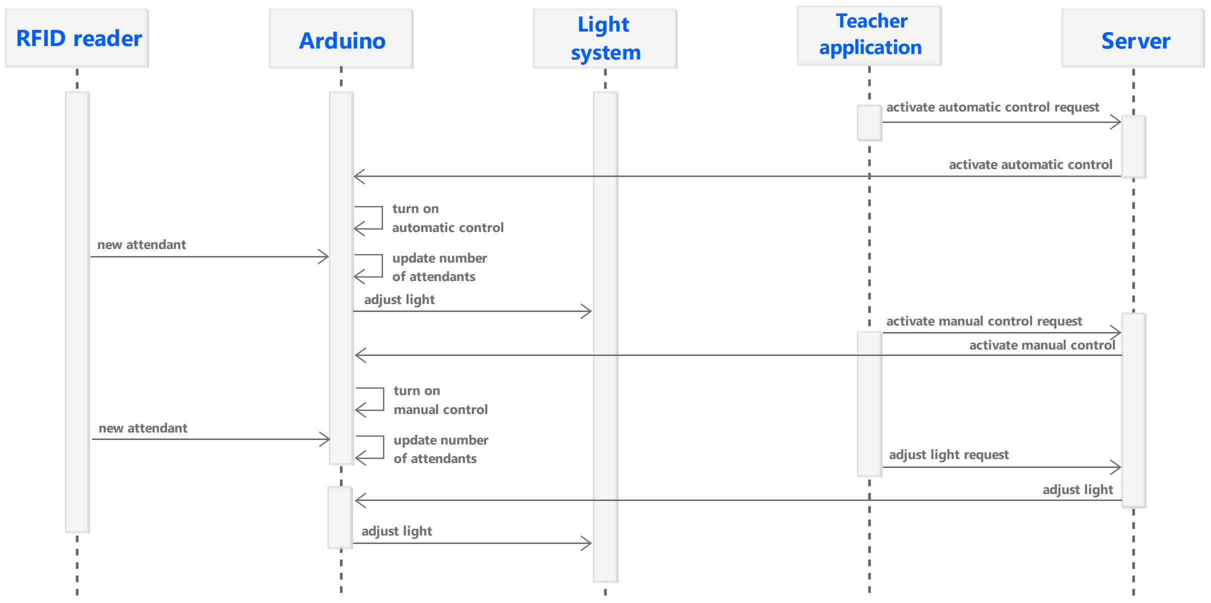


Figure 3.7: A sequence diagram for controlling light system.

As we see in *figure 3.7*, when the "Automatic control" is turned on, the Arduino adjusts the light system automatically. For "Manual control", the Arduino receives requests form the server and adjusts light system according them. In the manual modem, the system no longer depends on the number of attendants to adjust lights. However, the Arduino must keep updating number of attendants for another purpose which is checking attendant students.

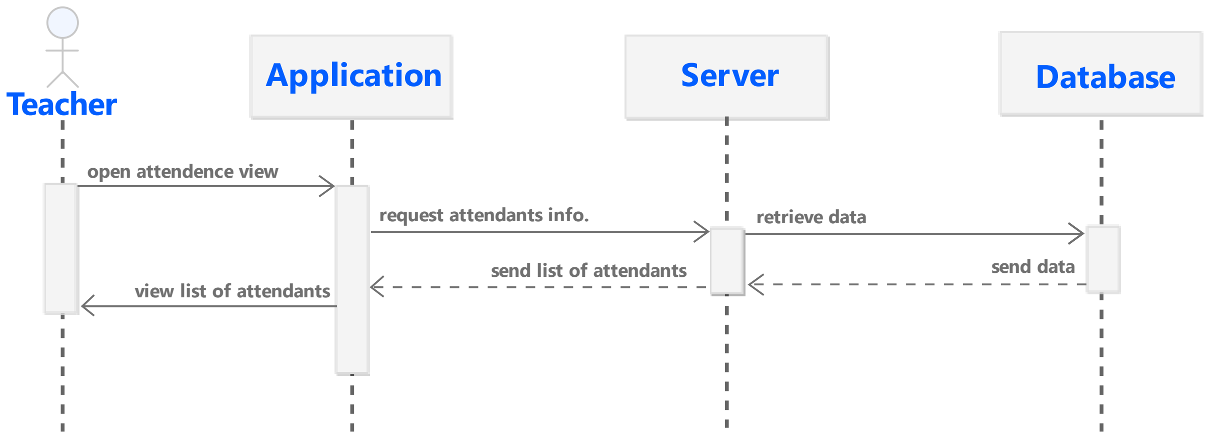


Figure 3.8: A sequence diagram for checking attendance.

The database contains information about attendant students. Keep in mind that once a student is authorized, he/she is marked as an attendant in the database. So, whenever the teacher wants to check attendance, the server will request the data of students that are marked from the database and sends it to the teacher application.

3.5 Tools and Languages

The use case and sequence diagrams gave a detailed description of the system, its requirements, and relations between different its parts. In this section we will provide information about the tools we will use to build the SLS. The section is divided into two subsections. First, we will discuss the tools needed to develop the hardware components. Then, we will discuss the needed languages, IDEs, libraries, and other software applications needed to design, program, and test our system.

3.5.1 Hardware Development Tools

We previously discussed the hardware components used in our system in *chapter 2* with a brief description about each component. These components are RFID reader/tag, Arduino board, relay module, and Ethernet module. These components need to be connected together and programmed to achieve the operations they are intended to do. In *chapter 5*, we will view how these components will be connected together. The following list shows the needed tools for the hardware development and a brief description about their usages:

1. **Proteus** – it is used to create schematics and electronic prints for manufacturing printed circuit boards.
2. **Arduino IDE** – it is used to write and upload programs to Arduino compatible boards
3. **Arduino Programming Language** – it is a simple programming language that is very similar to C language.

3.5.2 Software Development Tools

Developing our software application can be divided into main three parts, which are the design part, the implementation part, and the testing part. The design part involves designing diagrams and designing user interface of the mobile application. The implementation part involves programming languages, IDEs, frameworks, and libraries. The testing part involve server simulator. The following list shows the needed tools for the software development and a brief description about their usages:

1. **Software Ideas Modeler** – it is used to draw the UML diagrams.
2. **Adobe XD** – it is used to design user interfaces and prototypes.
3. **Android Studio** – it is an IDE to build mobile application for Android OS.
4. **RECT Native** – it’s a cross-platform that is used to build both Android and iOS based mobile applications.
5. **WAMP server** – it is software stack consisting of the Apache web server, OpenSSL for SSL support, MySQL database and PHP programming language.
6. **PHP storm –** it is an IDE designed for PHP programming language.
7. **Node JS** – it is a server-side programming language.
8. **Android smart phone –** it is a smart phone with Android OS installed on it.
9. **iPhone –** it can be considered as Apple’s smart phone.
10. **Google Chrome** – Google’s web browser.
11. **Redux** – It is an open-source JavaScript library for managing application state. It is most commonly used with libraries such as React or Angular for building user interfaces.

3.6 Summary

In this chapter we provide the reader with detailed knowledge about our system functional and non-functional requirements, user requirements. We provide UML diagrams that shows the details of how the system will function. In the end of the chapter we listed the needed tools for us to build the system.

Chapter 4: System Design

4.1 Introduction

In the previous chapter, we determined the different requirements of the SLS. We also determined the needed tools to build it. Now, we can design the system considering the determined requirements. In *section 4.2,* we provide the class diagram which includes the classes of the system, their attributes, operations, and relations. Then we provide the design of database tables in *section 4.3*. The design of the mobile application is included in *section 4.4*.

4.2 Class Diagram

A class diagram describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. *Figure 4.1* shows the class diagram that include classes representing all software/hardware components.

4.2.1 Teacher and Student

In our system each user can be either a teacher or a student. Both have some common attributes and operations. The common operations are grouped into a superclass which is called “User”. The two subclasses “Student” and “Teacher” will inherit the attributes and operation of their superclass in addition to their own distinctive attributes and operations.

4.2.2 Question

Since each student can ask a *one or more* questions, their will be a “Question” class contains distinctive id for each instance (question), the text of the question itself, the id of the student who submitted the questions, and the number of likes for the question.

4.2.3 Quiz and Result

The teacher needs to be able to create quizzes. To create a quiz, both the questions and corresponding answers should be provided to the system. Each instance of the “QuizQuestion” class represents a question and it’s given answers. The class “Quiz” *composites* an array or list of quiz questions.

SLS is designed to show results once for all student. The “QuizResult” class instances hold the id of the quiz, the id of each student, and each student degree. To set the degree of the student, the class have an operation that receives the answers and evaluates them considering the quiz id.

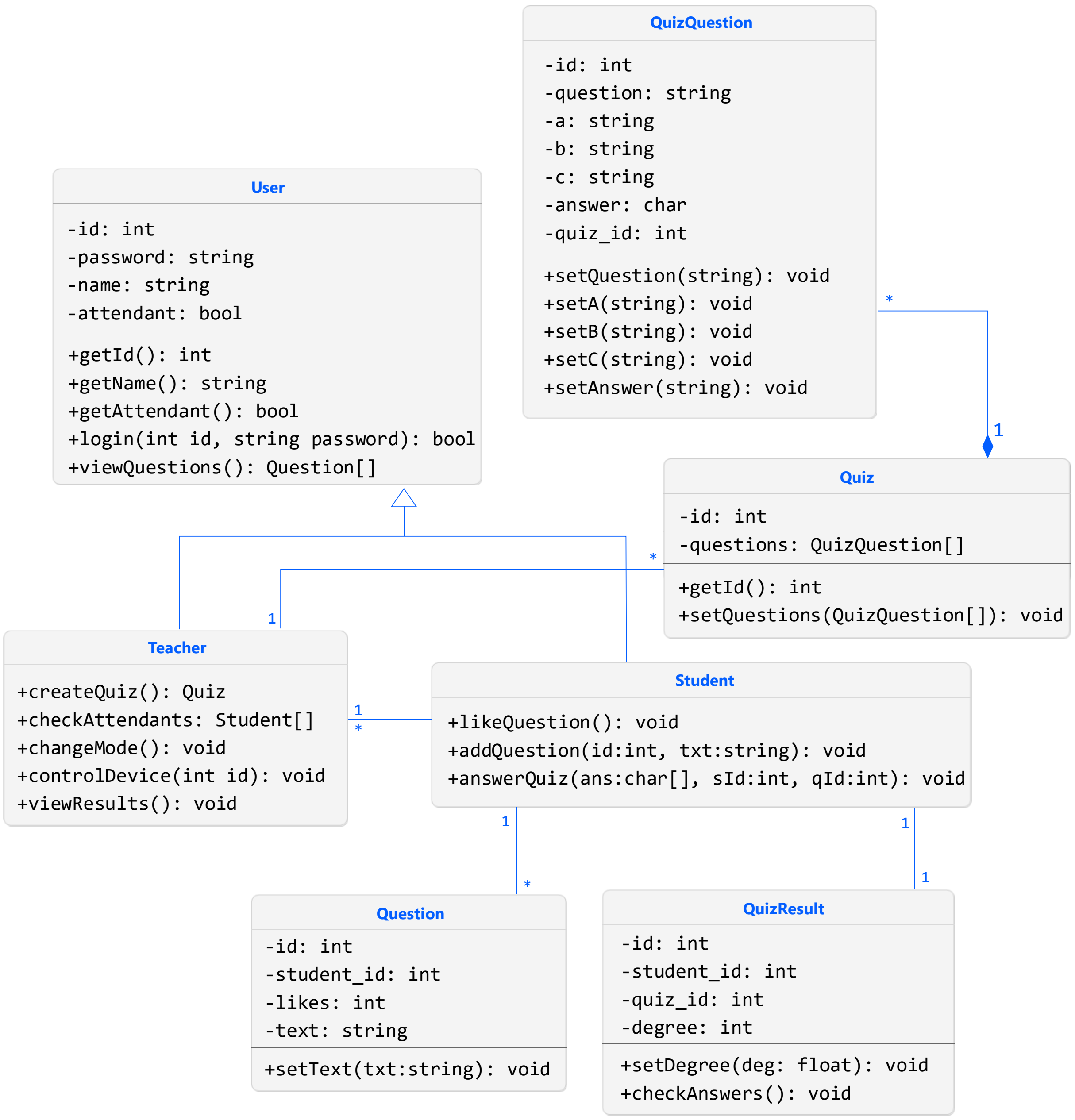
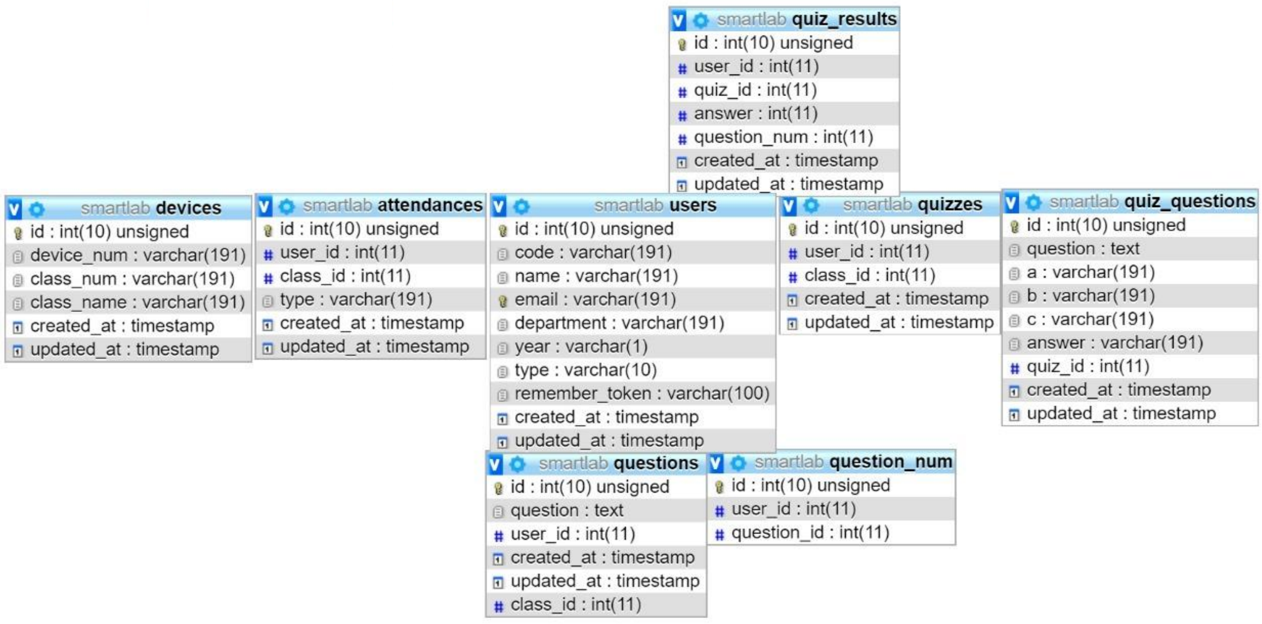


Figure 4.1: Class diagram

4.3 Database Design

In this section we will determine what data must be stored. With this information, we can begin to fit the data to the database model. *Figure 4.2* shows the tables and attributes for the database and the keys of each table.



smartlab quiz_results
- id : int(10) unsigned
- user_id : int(11)
- quiz_id : int(11)
- answer : int(11)
- question_num : int(11)
- created_at : timestamp
- updated_at : timestamp

smartlab devices
- id : int(10) unsigned
- device_num : varchar(191)
- class_num : varchar(191)
- class_name : varchar(191)
- created_at : timestamp
- updated_at : timestamp

smartlab attendances
- id : int(10) unsigned
- user_id : int(11)
- class_id : int(11)
- type : varchar(191)
- created_at : timestamp
- updated_at : timestamp

smartlab users
- id : int(10) unsigned
- code : varchar(191)
- name : varchar(191)
- email : varchar(191)
- department : varchar(191)
- year : varchar(1)
- type : varchar(10)
- remember_token : varchar(100)
- created_at : timestamp
- updated_at : timestamp

smartlab quizzes
- id : int(10) unsigned
- user_id : int(11)
- class_id : int(11)
- created_at : timestamp
- updated_at : timestamp

smartlab quiz_questions
- id : int(10) unsigned
- question : text
- a : varchar(191)
- b : varchar(191)
- c : varchar(191)
- answer : varchar(191)
- quiz_id : int(11)
- created_at : timestamp
- updated_at : timestamp

smartlab questions
- id : int(10) unsigned
- question : text
- user_id : int(11)
- created_at : timestamp
- updated_at : timestamp
- class_id : int(11)

smartlab question_num
- id : int(10) unsigned
- user_id : int(11)
- question_id : int(11)

Figure 4.2: Database tables and attributes.

4.4 Interface Design

A smartphone is required for each user to be able to use the system. We seek to minimize time and efforts needed by the user to do some action. So, the interface of the application should be clear, easy to use, easy to understand, and smooth. Each page in the application should focus on doing only on job. The following subsections show the designed views of the mobile application and description of each view.

4.4.1 Profile

After login successfully, the application will open the user's profile. The profile page contains user information, and logout button on the top. Below them is the options that represents interactions between users and teacher. Only for teacher profile, there are two options for checking attendence. These two options are designed to have different shape and color to *visually separate* them from other options and let the teacher find/access quickly. *Figure 4.3* shows two examples of teacher profile on the right and student profile on the left.

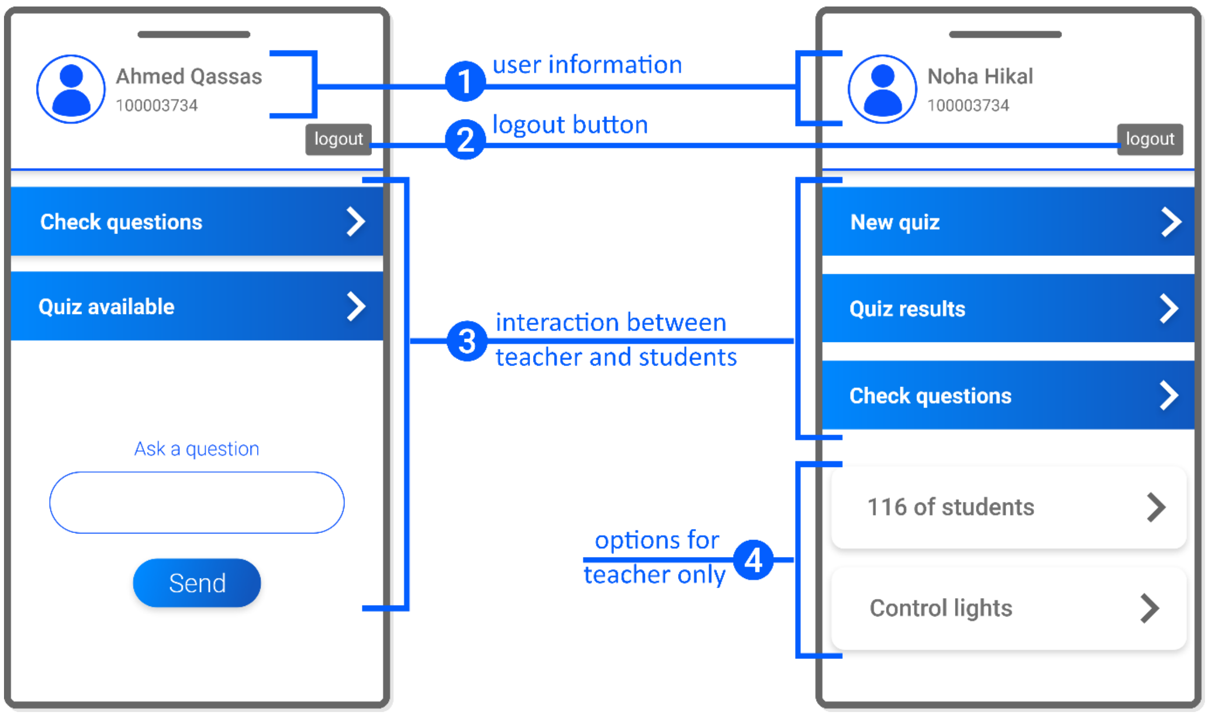


Figure 4.3: Student profile (on the left) and teacher profile (on the right).

4.4.2 Creating and Answering Quizzes

When the teacher wants to create a quiz, he/she can use the create quiz option. The "New Quiz" view contains one textbox that fits the whole screen width. Below it, three textboxes with smaller width for the choices. On the right to each one of them there is a radio button to indicate the right answer. At the bottom of the view, there are two buttons. The button on the left is used to add new question to the quiz. Once it's clicked the current question will be stored, and textboxes will be cleared for the new question. The button in the right, "Create", is used to create the quiz. Once it's clicked, the quiz should be sent to all attendants and the teacher should be redirected to the his/her profile. *Figure 4.4* shows the design of the "New Quiz" view.

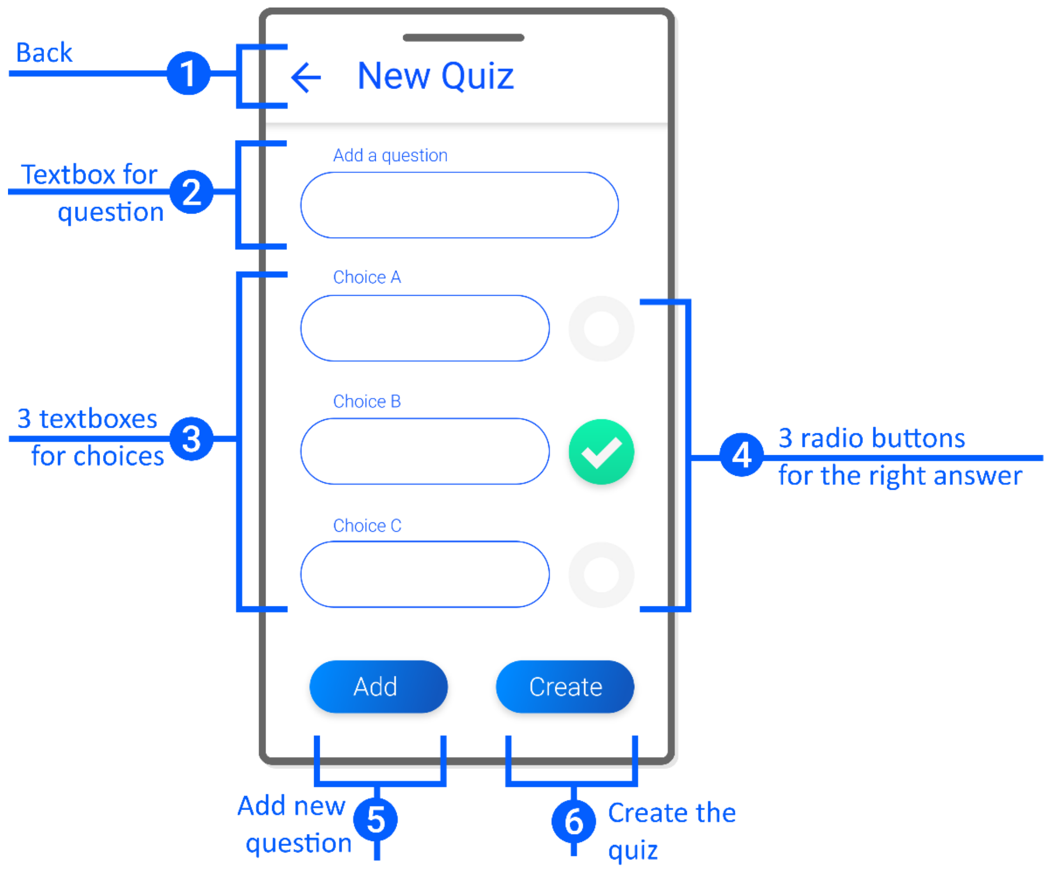


Figure 4.4: The view of creating a quiz.

Now a student can answer the quiz using his application. *Figure 4.5* shows how the quiz is displayed to student. The results of the quiz will be sent to the teacher once it is ready. The “Quiz Result” view consist of cards listed vertically. Each card represents a student and contains his/her name and degree. *Figure 4.6* shows the design of this view.

4.4.3 Control light and Check Attendance

The view that is designed to let the teacher control light system consist of a list of “Switch” options. The first switch allows the teacher to turn on or of the Automatic mode. When the Automatic mode is on, all the other switches are disabled. For the teacher to control the light system manually, he/she could turn off the Automatic mode switch and control each row individually. *Figure 4.7* shows how the view is designed. The view designed for checking attendants is very simple. *Figure 4.8* shows it.

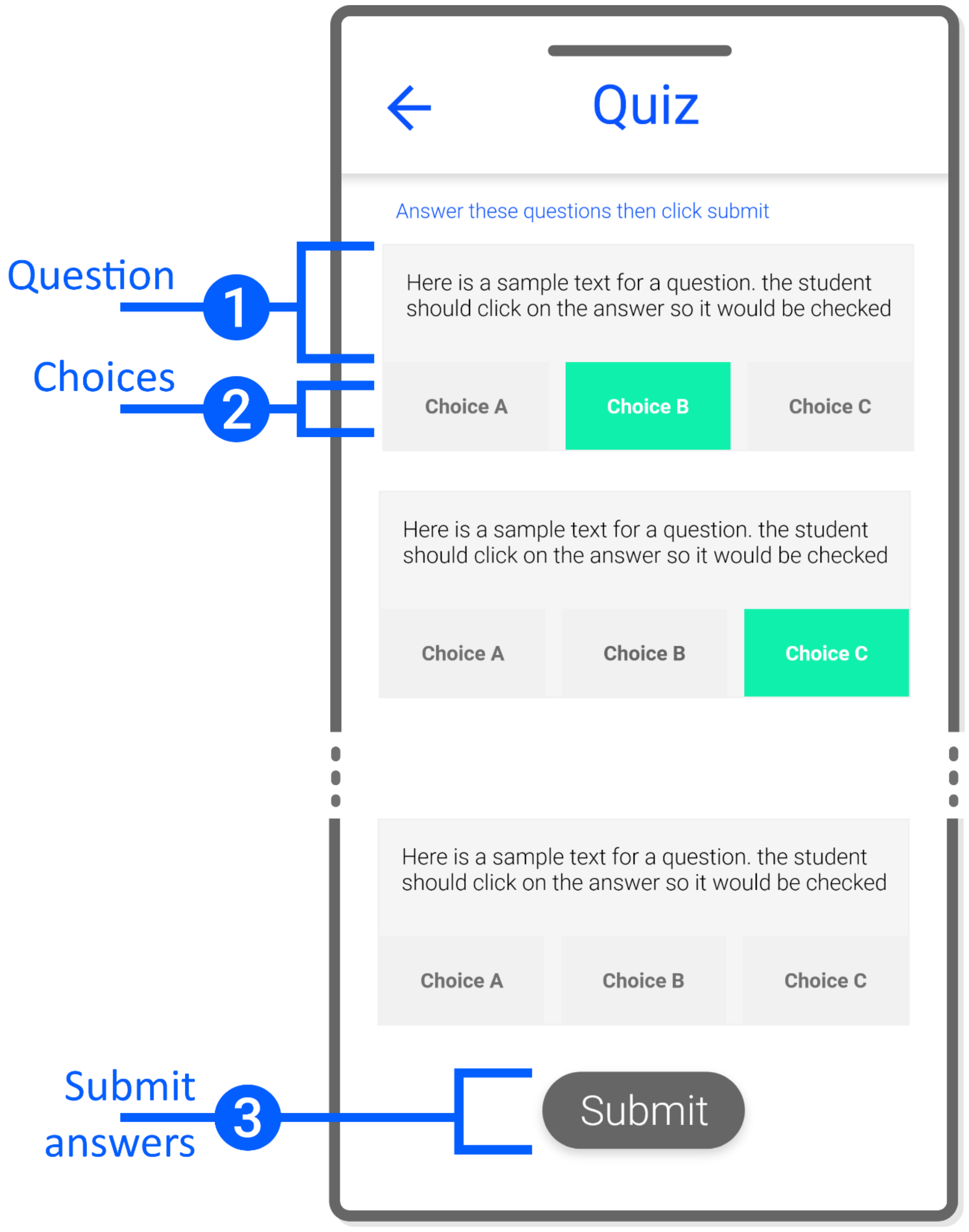
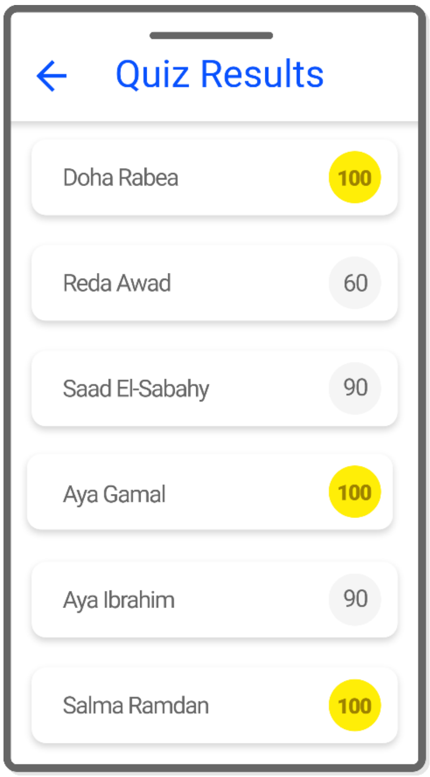

Quiz
Answer these questions then click submit
Question
1
Here is a sample text for a question. the student should click on the answer so it would be checked
Choices
2
Choice A
Choice B
Choice C
Here is a sample text for a question. the student should click on the answer so it would be checked
Choice A
Choice B
Choice C
Here is a sample text for a question. the student should click on the answer so it would be checked
Choice A
Choice B
Choice C
Submit answers
3
Submit

Figure 4.5: Displaying the quiz to students



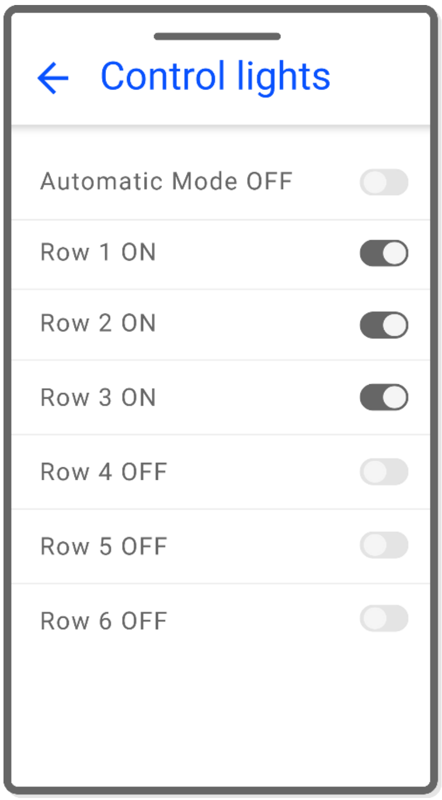
Quiz Results

Doha Rabea	100
Reda Awad	60
Saad El-Sabahy	90
Aya Gamal	100
Aya Ibrahim	90
Salma Ramdan	100

Figure 4.6: Quiz Results view.

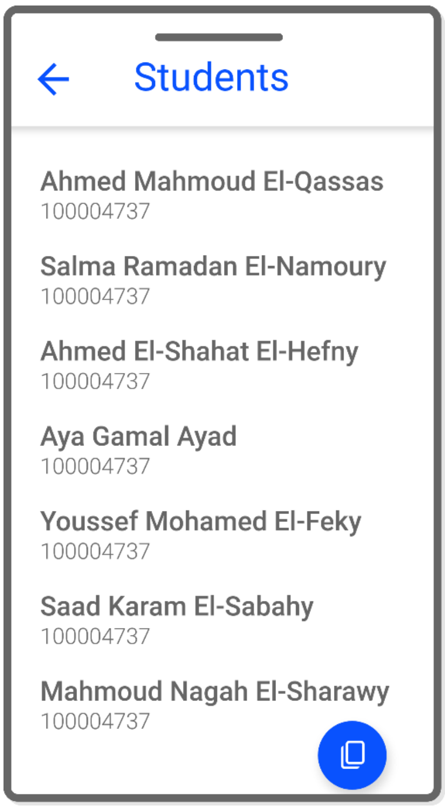
Figure 4.7: Control lights view.

Figure 4.8: Attendants view.



Control lights

Automatic Mode OFF	off
Row 1 ON	on
Row 2 ON	on
Row 3 ON	on
Row 4 OFF	off
Row 5 OFF	off
Row 6 OFF	off



Students

Ahmed Mahmoud El-Qassas
100004737

Salma Ramadan El-Namoury
100004737

Ahmed El-Shahat El-Hefny
100004737

Aya Gamal Ayad
100004737

Youssef Mohamed El-Feky
100004737

Saad Karam El-Sabahy
100004737

Mahmoud Nagah El-Sharawy
100004737

4.4.4 Student Questions

The view that is designed to display questions is represented as shown in *Figure 4.9*. The view is very similar for both students and teacher. However, A teacher has no need to like a question, so this button won't appear in the view of the teacher. Each card contains question and name of the student who asked it. Below each card, the number of students who liked the question is provided.

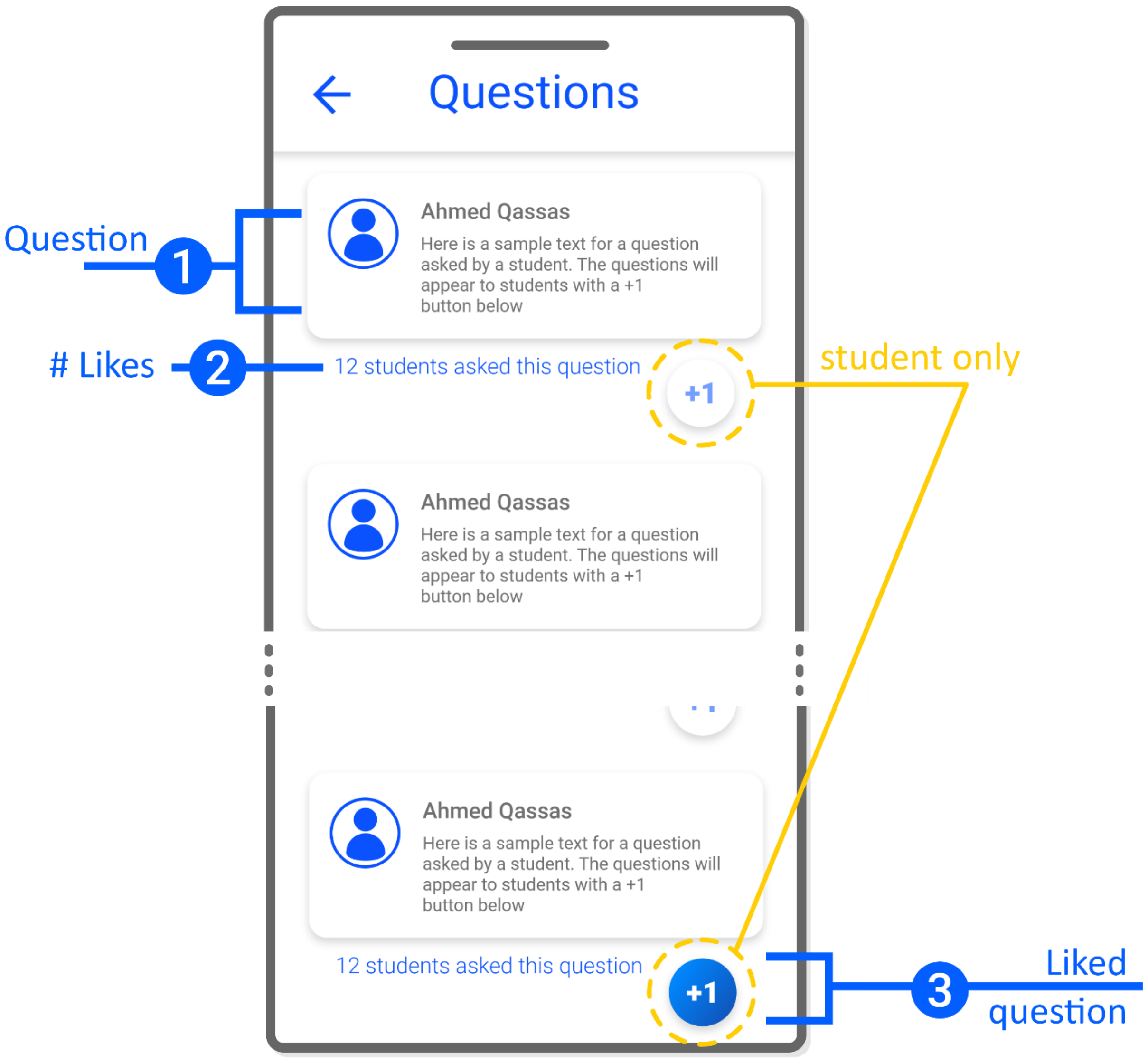

Questions
Question
1
# Likes
2
student only
Liked question
3
Ahmed Qassas
Here is a sample text for a question asked by a student. The questions will appear to students with a +1 button below
12 students asked this question
+1

Figure 4.9: The view used to display students' questions.

4.5 Summary

In this chapter, we designed the class diagram for the SLS. We also designed the tables of the database. And, we showed the reader the interface design of the application with a description of its usage. We can now start the implementation process. In the next chapter, we will map our system design to implementation.

Chapter 5: System Implementation

5.1 Introduction

After we designed the system, we will map our design into implementation. *Section 5.2* discusses the implementation process considering our objectives. Next in *Section 5.3*, we view samples of the code. *Section 5.4* and *section 5.5* discuss the testing process and result respectively.

5.2 Mapping Design to Implementation

The process of implementing our design is divided into three parts. The first part is the configuration of hardware components. The second part is building the mobile application. The third part is the integration between the hardware and the mobile applications.

There are two roles for the HW components used in the system. The first role is identifying the user. The second role is to control the light system. A demonstration how to implement these objectives below:

A. RFID (Identification Method)

As we mentioned in *Chapter 2*, RDIF technology depends on the electromagnetic waves produces by an antenna. The electromagnetic waves induce a circuit inside the RFID Tag, which produces electric current sufficient for the Tag to release the information stored in its memory. This kind of Tags is called passive (contain no battery). It's not expensive like active RFID Tags and appropriate for our system.

B. Arduino

Arduino role will be controlling the light system and counting the number of identified users. The Arduino needs other components to operate. *Chapter 2* contains information about these components. *Figure 5.1* shows how to connect Arduino and RFID module together.

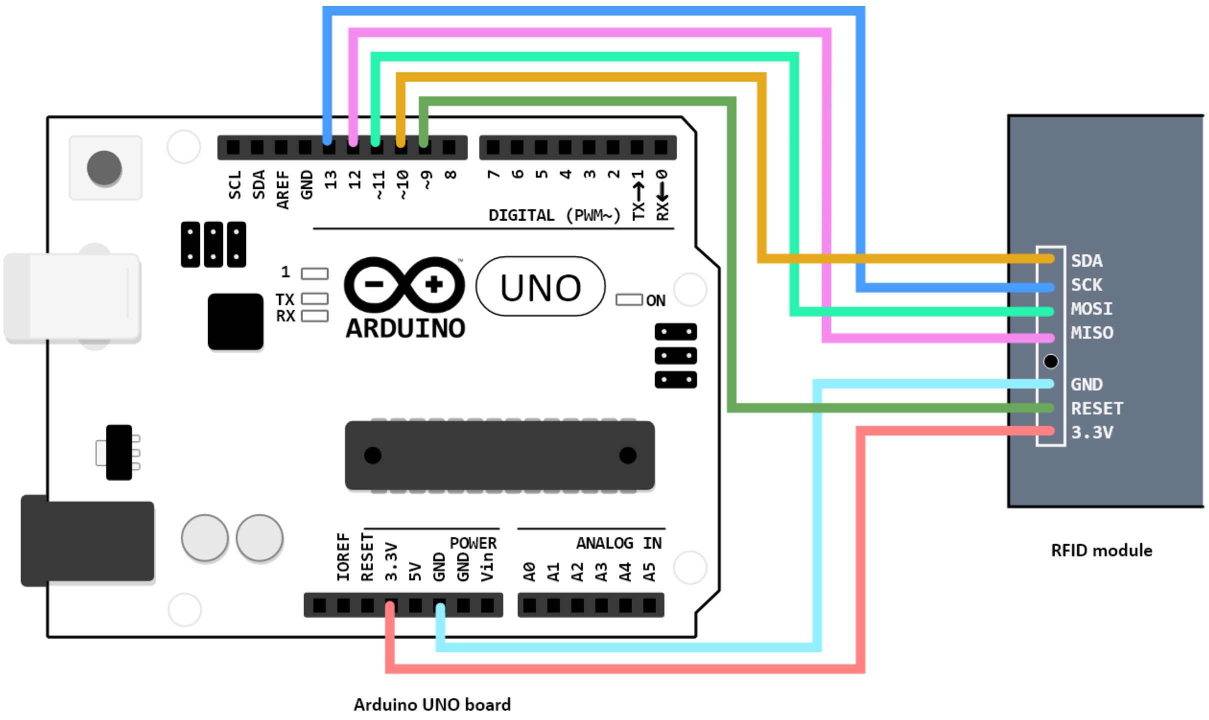


Figure 5.1: connecting Arduino board and RFID together

However, this is not how the connection actually implemented; as there are two other components needs to be connected to the Arduino which are Relay module and Ethernet module. We will first use test boards for connecting HW components together. Then, when everything is running well, we build the Printed Circuit Board (PCB). *Figure 5.2* and *Figure 5.3* show the real implementation connections using test board and PCB respectively.

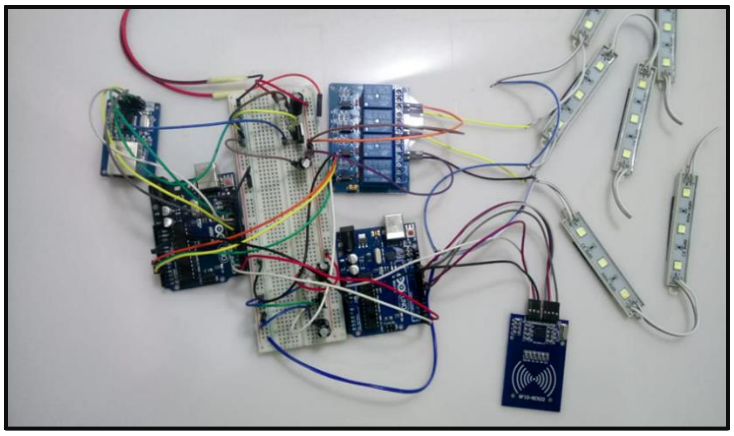


Figure 5.2: HW connections using test boards.

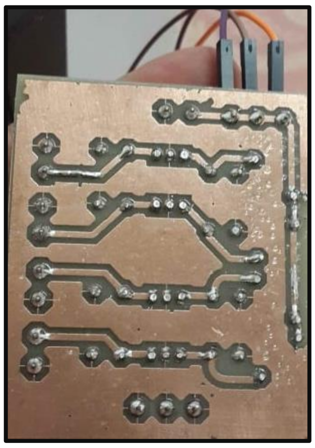
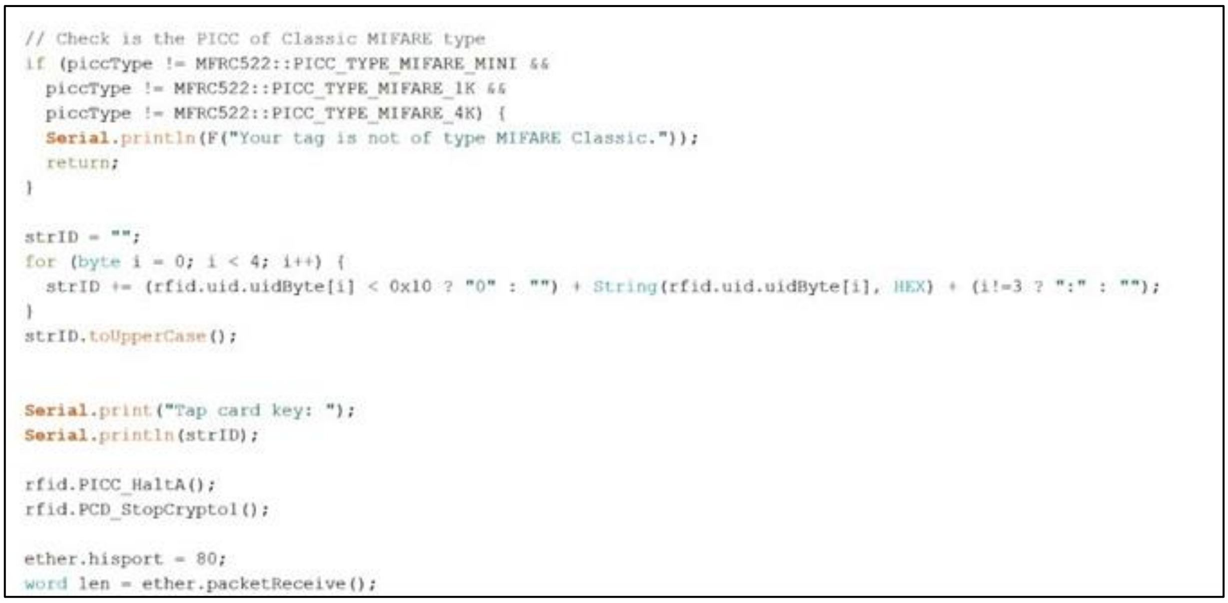


Figure 5.3: PCB

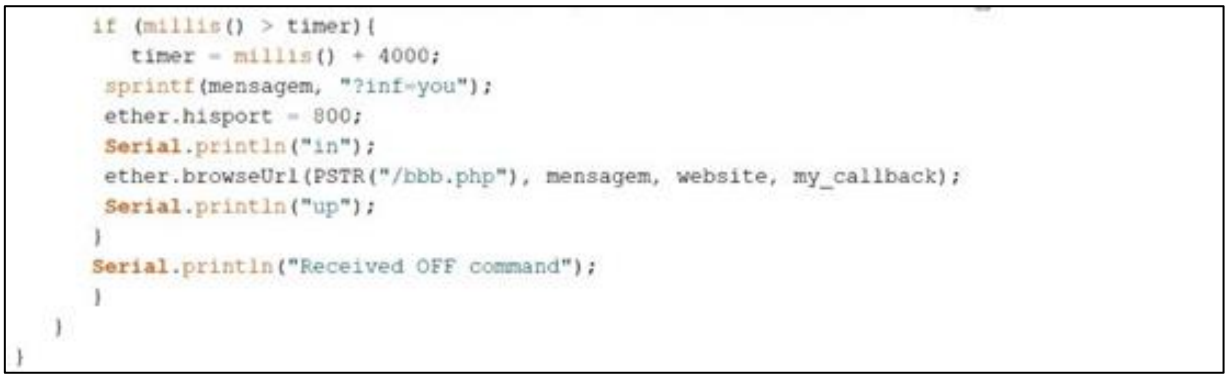
5.3 Code Samples

Connecting HW components is not the only step. For these components to achieve their goal, Arduino is to be programmed using the Arduino IDE. The following is code used to let the Arduino read the Id of the RFID Tag.



```
// Check is the PICC of Classic MIFARE type
if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
  piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
  piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
  Serial.println(F("Your tag is not of type MIFARE Classic."));
  return;
}

strID = "";
for (byte i = 0; i < 4; i++) {
  strID += (rfid.uid.uidByte[i] < 0x10 ? "0" : "") + String(rfid.uid.uidByte[i], HEX) + (i!=3 ? ":" : "");
}
strID.toUpperCase();


Serial.print("Tap card key: ");
Serial.println(strID);

rfid.PICC_HaltA();
rfid.PCD_StopCrypto1();

ether.hisport = 80;
word len = ether.packetReceive();
```

We used HTTP protocol to exchange data over the network. The code below is used to send Id over the network.



```
      if (millis() > timer){
         timer = millis() + 4000;
       sprintf(mensagem, "?inf=you");
       ether.hisport = 800;
       Serial.println("in");
       ether.browseUrl(PSTR("/bbb.php"), mensagem, website, my_callback);
       Serial.println("up");
      }
      Serial.println("Received OFF command");
      }
   }
}
```

The code below for interface design will be rendered to the view “New Quiz”

```
import React ,{Component} from 'react';
import {View,ScrollView} from 'react-native';
import CreateQuizC  from './public/CreateQuizC'
export default class  CreateQuiz extends Component{
    render(){
        const{container_fluid}=this.styles;
        return(
        <ScrollView>
            <View style={container_fluid}>
                <CreateQuizC/>
            </View>
        </ScrollView>
        )
    }
    styles={
        container_fluid:{
            flex:1,
            height:"100%",
            padding:20,
            backgroundColor:'#fff'
        },
    }
}
```

The code below for interface design will be rendered to “Quiz Result” view

```
import React from 'react';
import {View,Text} from 'react-native';
const QuizResC=(props)=>{
    const{container,text,text1,yellowC}=styles;
    return(
        <View style={container}>
```

```
            <View>
                <Text style={text}>{props.name}</Text>
            </View>
            <View
style={props.mark==props.fullMark?{...text1,...yellowC}:text1}>
                <Text                    style={text}>{props.mark
+'/'+props.fullMark}</Text>
            </View>
        </View>
    )
}
const styles={
    container:{
        width:"100%",
        flexDirection:'row',
        justifyContent:'space-between',
        alignItems:'center',
        marginTop:30,
        paddingLeft:20,
        paddingRight:20,
        paddingTop:10,
        paddingBottom:10,
        borderWidth:0,
        borderRadius:16,
        shadowColor:'#000',
        shadowOffset:{
            width:0,
            height:2,
        },
        shadowOpacity:0.2,
        shadowRadius:5,
        elevation:1
    }  ,
    text:{
        fontSize:25,
        color:"#757575",
        "fontWeight":"500"
    } ,
    text1:{
        backgroundColor:'#F5F5F5',
        fontWeight:'bold',
        color:'brown',
        fontSize:18,
        padding:10,
        width:70,
        height:70,
        borderRadius:35,
        borderColor:'#adadad',
        position:'relative',
        alignItems:'center',
        justifyContent:'center',
    },
    yellowC:{
        backgroundColor:"#FFEE07",
    } ,
};
export default QuizResC;
```

The code below is for “Login” view.

```
import React, {Component} from 'react';
import {View,Image} from "react-native";
import logo from "./assets/images/logo.png";
import {Btn, Input} from "./common";
import {connect} from "react-redux";
import {IdChange,SignUp,setUsers} from "../actions";
```

```
class AuthStudent extends Component {
    onCodeChange=(text)=>{
        this.props.IdChange(text);
    };
    SignUp=()=>{
        this.props.SignUp({code:this.props.id});
    };
    render() {
        const {
            mainContainerStyle,
            logoContainerStyle,
            logoStyle,
            containerBtnStyle,
            inputContainerStyle,
            buttonStyle
        }=styles;
        return (
            <View style={mainContainerStyle}>
                <View style={logoContainerStyle}>
                    <Image source={logo} style={logoStyle} />
                </View>

                <View style={inputContainerStyle}>
                    <Input
                        placeholder="Submit Your Id"
                        headerTextValue="Sign Up"
                        onChangeText={this.onCodeChange}
                        value={this.props.id}
                    />
                </View>
                <View style={containerBtnStyle}>
                        <Btn         children={"Sign         Up"}
loading={this.props.loading} onPress={this.SignUp}
```

```
                        style={buttonStyle}/>
                </View>
            </View>
        );
    }
}
const styles={
    mainContainerStyle:{
        display:"flex",
        flex:1,
        backgroundColor:'#fff',
        flexBasis:'100%',
    },
    logoContainerStyle:{
        display:"flex",
        flex:2,
        marginTop:"10%",
        marginBottom:"20%"
    },
    logoStyle:{
        width:"100%",
    },
    inputContainerStyle:{
        flex:2,
       width:'80%',
        alignSelf: 'center'
    },
    inputStyle:{
        padding: 10
    },
    containerBtnStyle:{
        flex:5,
        alignSelf:"center",
        width: "60%",
        marginTop: 30
    },
    /*buttonStyle:{
        padding:22
    },*/
    buttonTextStyle:{
        fontSize:23,
        fontWeight: '400'
    },
};
const mapStateToProps=(state)=>{
    return {
        id:state.Auth.id,
        loading:state.Auth.loading,
        user:state.user
    }
};
export                                                        default
connect(mapStateToProps,{IdChange,SignUp,setUsers})(AuthStudent);
```

The code below is the backend development for quizzing process.

```
<?php
namespace App\Http\Controllers\quiz;
use App\Attendance;
use App\Http\Resources\quizResultResource;
use App\Quiz;
use App\QuizQuestion;
use App\QuizResult;
use App\User;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
```

```
class quizController extends Controller
{
    public function store(Request $request){
        $user=User::find($request->user_id);
        $class=Attendance::where('user_id',$user->id)->first();
        $quiz=new Quiz();
        $quiz->user_id=$user->id;
        $quiz->class_id=$class->class_id;
        if($quiz->save())
        {
            $q=json_decode($request->quiz);
            if($q)
            {
                foreach ($q as $qu)
                {
                    $question=new QuizQuestion();
                    $question->question=$qu->question;
                    $question->a=$qu->a;
                    $question->b=$qu->b;
                    $question->c=$qu->c;
                    $question->answer=$qu->answer;
                    $question->quiz_id=$quiz->id;
                    $question->save();
                }
            }
            return      response()->json(['msg'=>'quiz      stored
successfully','type'=>'success'],200);
        }
        else
        {
            return      response()->json(['msg'=>'quiz      stored
failed','type'=>'error'],200);
        }
    }
    public function index(Request $request){
        $class=Attendance::where('user_id',$request->user_id)-
>first();
        if ($class)
        {
            $quiz=Quiz::where('class_id',$class->class_id)-
>get();
            return response()->json(['quizs'=>$quiz ?? []],200);
        }
        return  response()->json(['msg'=>"you  can't  access  to
quiz"],200);
    }
    public function show($id){
        $quiz=QuizQuestion::where('quiz_id',$id)->get();
        if ($quiz)
        {
            return response()->json(['quiz'=>$quiz],200);
        }
        return response()->json(['msg'=>"quiz not found"],200);
    }
    public function quizResult($id,Request $request){
        $answers=json_decode($request->answers);
        $ans=QuizResult::where('quiz_id',$id)-
>where('user_id',$request->user_id)->first();
        if(!$ans)
        {
            if($answers && count($answers)>0)
            {
                $answer=0;
                foreach ($answers as $item)
                {
                    $question=QuizQuestion::find($item->id);
                    if($question)
                    {
                        if($question->answer==$item->answer)
                        {
                            ++$answer;
                        }
                    }
                }
                $question_num=QuizQuestion::where('quiz_id',$id)-
>get();

                $res=new QuizResult();
                $res->user_id=$request->user_id;
                $res->quiz_id=$id;
                $res->answer=$answer;
                $res->question_num=count($question_num);
                $res->save();
                return    response()->json(['msg'=>"answer    sent
successfully",'type'=>'success'],200);
            }


            else
            {
                return   response()->json(['msg'=>"please   submit
answers",'type'=>'error'],200);
            }
        }
        return  response()->json(['msg'=>"you  can't  answer  this
quiz again",'type'=>'error'],200);
    }
    public function getQuizResult($id)
    {
        $res=QuizResult::where('quiz_id',$id)->get();
        if($res)
        {
            return                                          response()-
>json(['quizResults'=>quizResultResource::collection($res)],200);
        }
        return                          response()->json(['msg'=>"no
result",'type'=>'error'],200);
    }
}
```

```
import {
    LOADING,
    SEND_QUESTION,
    TEXT_CHANGE,
    RENDER_QUESTION,
    PLUS_COUNTER,
    DELETE_MESSAGE,
    SET_QUIZ,
    CURRENT_QUIZ,
    SET_RESULT,
    CHANGE_RESULT,
    SET_MSG
} from "./types";
import axios from '../axios';
import {Actions} from "react-native-router-flux";

export const TextChange=(text)=>{
    return({
        type :TEXT_CHANGE,
        payload:text,
    })
};
export const SendQuestion=({question,id})=>{

    return(dispatch)=>{
        dispatch({type:LOADING,payload:true});



axios.post('/question',{'question':question,'user_id':id}).then((
res)=>{
            if(!res.data.error)
            {
                dispatch({type:
SEND_QUESTION,payload:{question,id}});
                dispatch({type:LOADING,payload:false});
            }


        }).catch((rej)=>{
        })
    }
};
export const RenderQuestion=()=>{
    return(dispatch)=>{
        axios.get('/question').then((res)=>{

            dispatch({type
:RENDER_QUESTION,payload:res.data.questions});


        })
    }
};
export const PlusCounter=({QuestionId,UserId})=>{

    return(dispatch)=>{
        dispatch({type:DELETE_MESSAGE});
        axios.put(`/question/${QuestionId}`,{
'user_id':UserId}).then((res)=>{
            if (res.data.type==='success')
            {
                dispatch({type
:RENDER_QUESTION,payload:res.data.questions});
            }
            dispatch({type :PLUS_COUNTER,payload:res.data});


        })
    }
};
export const DeleteMessage=()=>{
    return{
        type:DELETE_MESSAGE,

    }

};
export const RenderQuiz=(id)=>{

    return(dispatch)=>{
        dispatch({type:DELETE_MESSAGE});

        axios.get('/quiz',{
            params:{
                'user_id':id
            }
        }).then((res)=>{
            dispatch({
                type:SET_QUIZ,
                payload:res.data.quizs
            })
        })
    }
};
export const oneQuiz=(id)=>{

    return(dispatch)=>{
        dispatch({type:DELETE_MESSAGE});

        dispatch({
            type:SET_RESULT,
            payload:[]
        });

        dispatch({
            type:CURRENT_QUIZ,
            payload:[]
        });


        axios.get('/quiz/'+id).then((res)=>{
            let q=res.data.quiz;
            if(q)
            {
                let r=[];
                q.forEach((item)=>{
                    r.push({id:item.id,answer:'a'})
                });
                dispatch({
                    type:SET_RESULT,
                    payload:r
                });

                dispatch({
                    type:CURRENT_QUIZ,
                    payload:q
                });
            }
        })
    }
};
export const changeResult=(res)=>{

    return({
        type :CHANGE_RESULT,
        payload:res,
    })

}
export const sendAnswer=({results,id,user_id})=>{

    return(dispatch)=>{
        dispatch({type:DELETE_MESSAGE});
        let formData=new FormData();
        formData.append('user_id',user_id);
        formData.append('answers',JSON.stringify(results));
        dispatch({type:LOADING,payload:true});
        axios.post('/quiz/'+id,formData).then((res)=>{
           console.log(res);
            dispatch({
                    type:SET_RESULT,
                    payload:[],
                });
            dispatch({type:LOADING,payload:false});

                dispatch({
                    type:CURRENT_QUIZ,
                    payload:null
                });
                dispatch({
                    type:SET_MSG,
                    payload:res.data
                });
                Actions.AllQuiz();
        }).catch((rej)=>{
            console.log(rej.response)
        })
    }

}<?php

namespace App\Http\Controllers;

use App\User;
use Illuminate\Http\Request;

class RegisterController extends Controller
```

```
{
    public function signup(Request $request)
    {
        $user=User::where('code',$request->code)-
>first(['code','type']);
        if($user)
        {
            return response()->json(['userData'=>$user],200);
        }
        return  response()->json(['msg'=>"you  can't  sign  up  with
this code",'type'=>'warn'],200);

    }
}
```

```
<?php
namespace App\Http\Controllers;
use App\Attendance;
use App\Device;
use App\question;
use App\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use App\Http\Controllers\Controller;

class AuthController extends Controller
{
    /**
     * Create a new AuthController instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth:api',         ['except'         =>
['login','deviceLogin','loginDoc']]);
    }
    /**
     * Get a JWT via given credentials.
     *
     * @return \Illuminate\Http\JsonResponse
     */
    public function login(Request $request)
    {
        $user = User::whereCode(request('code'))->first();
        $token=null;
        if($user)
        {
            $attend=      Attendance::where('user_id',$user->id)-
>first();

            if($user->type=='student')
            {
                if(!$attend)
                {
                    return   response()->json(['msg'   =>   'please
login             from             attendance             device
first','type'=>'warn','is_attend'=>false], 200);
                }
                else
                {
                    $doctor=Attendance::where('class_id',$attend-
>class_id)->where('type','doctor')->first();
                    if($doctor)
                    {
                        $token = auth()->login($user);
                        if (! $token) {
                            return   response()->json(['error'   =>
'Unauthorized'], 401);
                        }

                        return                             $this-
>respondWithToken($token,$doctor->class_num);
                    }
                    else
                    {
                        return response()->json(['msg' => 'please
wait    until    doctor    start    lecture    and    click    button
blue','type'=>'warn','is_attend'=>true], 200);

                    }
                }

            }
        }
        else
        {
            return     response()->json(['msg'     =>     'invalid
user','type'=>'warn'], 200);

        }

    }
    public function loginDoc(Request $request)
    {
        $user = User::whereCode(request('code'))->first();
        $token=null;
        if($user)
        {
            if($user->type=='doctor')
            {
                $attend=   Attendance::where('user_id',$user->id)-
>first();


                if(!$attend)
                {
                    $class=Device::where('class_num',$request-
>class_num)->first();
                    $attend=new Attendance();
                    $attend->user_id=$user->id;
                    $attend->class_id=$class->id;
                    $attend->type='doctor';
                    $attend->save();
                }

                $token = auth()->login($user);
                if (! $token) {
                    return      response()->json(['error'      =>
'Unauthorized'], 401);
                }

                return $this->respondWithToken($token);

            }
            else
            {
                return response()->json(['msg' => "you can't login
with this code",'type'=>'warn'], 200);

            }

        }
        else
        {
            return     response()->json(['msg'     =>     'invalid
user','type'=>'warn'], 200);

        }

    }

    public function deviceLogin(Request $request)
    {
        $user = User::whereCode($request->code)->first();
        if($user)
        {
           $attend=       Attendance::where('user_id',$user->id)-
>first();
           $class=Device::where('class_num',$request->class_num)-
>first();
           if(!$attend){
               $attend=new Attendance();
               $attend->user_id=$user->id;
               $attend->class_id=$class->id;
               $attend->type='student';
               $attend->save();
           }
        }
        $number=Attendance::where('class_id',$class->id)-
>count();
        return response()->json(['number'=>$number]);
    }

    /**
     * Get the authenticated User.
     *
     * @return \Illuminate\Http\JsonResponse
     */
    public function me()
    {
        return response()->json(auth()->user());
    }

    /**
     * Log the user out (Invalidate the token).
     *
     * @return \Illuminate\Http\JsonResponse
     */
    public function logout(Request $request)
    {
        $class=Device::where('class_num',$request->class_num)-
>first();
        Attendance::where('class_id',$class->id)->delete();
        question::where('class_id',$class->id)->delete();
        return  response()->json(['msg'  =>  'Successfully  logged
out']);
    }

    /**
     * Refresh a token.
     *
     * @return \Illuminate\Http\JsonResponse
     */
    public function refresh()
    {
        return $this->respondWithToken(auth()->refresh());
    }

    /**
     * Get the token array structure.
     *
     * @param  string $token
     *
     * @return \Illuminate\Http\JsonResponse
     */
    protected function respondWithToken($token,$class_num=null)
    {
        return response()->json([
            'access_token' => $token,
            'token_type' => 'bearer',
            'user'=>auth()->user(),
            'expires_in' => auth()->factory()->getTTL() * 60,
            'class_num'=>$class_num
        ]);
    }
}
```

If the reader wants to view the full code of the project, it is provided on the DVD. See *Appendix A* to know the contents and the file structure of the DVD and how to use it.

5.1 System Testing

When you are doing something for the for the first time, it will probably be not perfect and there should be a little, big, or fatal mistake in it. So, there is a great need to test our system and see how it will operate in all cases. Testing is the testing of a complete and fully integrated software product. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system. White box testing is the testing of the internal workings or code. In contrast, black box or System Testing is the opposite. System test involves the external workings of the software from the user's perspective. The table, *Table 5.1*, below represents the white testing for the HW components. Table 5.2 show test cases for the whole system.

#	Test Case	Expected result	Actual result
1	For the relay module: - Vcc = 5v - GND (ground) - N1 is on - N2 is on - N3 is off - N4 is off	The module should operate. The first and second **(only)** rows of LEDs should be turned on.	Success
2	For the RFID module provided with (Valid) RFID Tag and Vcc = 3.3v	RFID identifies the tag and Tag id is extracted	Success
3	The RFID module provided with (invalid) RFID Tag. Vcc = 3.3v	RFID module takes no actions.	Success
4	For the relay module: - Vcc > 5v - GND (ground) - N1 is on - N2 is on - N3 is on	The module will not run and may burn.	Test couldn't be done due to financial reasons
5	For the relay module: - Vcc < 5v - GND (ground) - N1 is on - N2 is on - N3 is on	The module will not run	Success

Table 5.1: White testing for HW components.

#	Test Case	Expected result	Actual result
1	Sending request for application while the Ethernet module is supplied with 5v and connected to a network with RJ45 cable.	Request is sent	Success
2	A user provides a valid RFID Tag to the reader and logs in with correct username and password.	User logs in successfully and the user profile page will be available. The user is marked as an attendant in the database. New LEDs row should be turned on (if needed).	Success
4	A user provides a valid RFID Tag to the reader and logs in with incorrect username or password.	User can't login. The user is not marked as attendant in the database. New LEDs row should be turned on (if needed).	Success
5	A teacher creates a quiz.	Quiz is sent to all students	Success
6	All students answered the quiz. Time **is not** out.	Quiz results appears to the teacher	Success
7	99% of students answered the quiz and 1% didn't submit the answer. The time **is not** out.	No actions are taken.	Success
8	99% students answered the quiz and 1% didn't answer it. Time is out.	Quiz results is sent to the teacher.	Success

Table 5.2: Black box testing

5.2 Results

The operations provided by our system, SLS, helps the teacher and students to focus on important activities. SLS, also can save power consumption. For example, given a classroom with 100 light bulbs distributed in rows with 10 bulbs for each row. Typically, in normal environment without our system, either all the 100 bulbs are turned on or you had to control the yourself physically. The both options waste power or time. With SLS installed in such a classroom with only 70% of it is occupied, you can save 30% of power consumption. And in case you need to turn on/off specific or all rows, you can do this throw the mobile application with clear, simple UI. Figure 5.4 illustrates the given example.

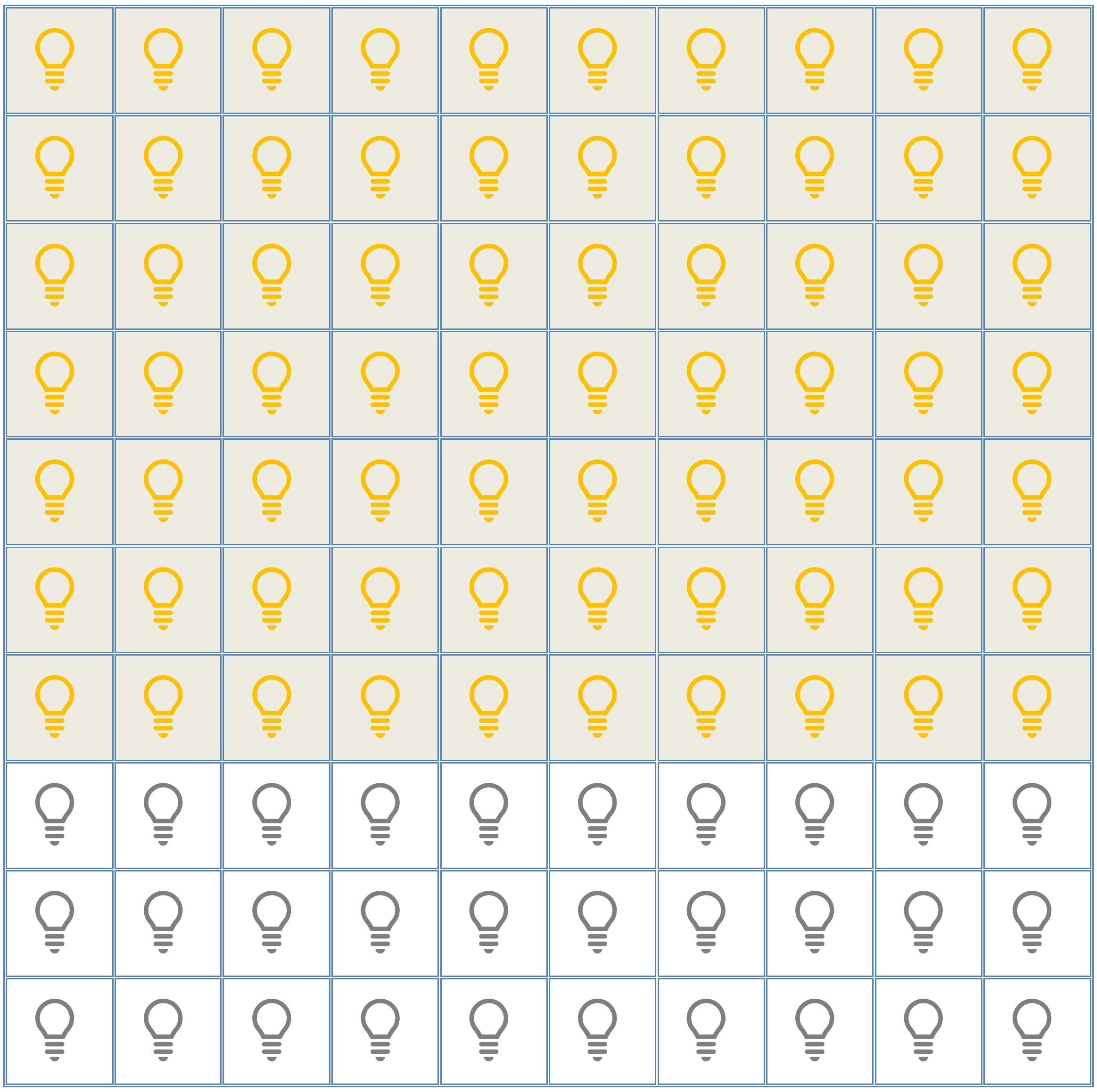


Figure 5.4: How SLS reduces power consumption.

5.3 Goals Achieved

We managed to build our system using simple components. The system achieved all its basic goals. The system controls the light system automatically. The system is able to check attendance automatically. The system allows the teacher to create quizzes and evaluates them automatically. Doing all these processes automatically will save great effort. the student can submit questions (or feedbacks) using the application which gives the teacher an idea about the students' understanding, the hard parts for them or even conclude the best practice for them to understand lessons. The teacher can view attendants using the application. However, exporting these data into excel sheet is not complete it due to time reasons. Table 5.3 shows the achieved objectives mentioned in *Chapter 1*.

#	Goal	Achieved	Notes
1	To adjust the lights in a classroom automatically or manually.	Yes	_
2	To digitize the process of assigning and evaluating quizzes.	Yes	_
3	To allow students to submit their questions or feedback using smartphones.	Yes	_
4	To check student attendance automatically.	Yes	The system checks attendance automatically. More time is needed to exporting data as excel sheet will be considered in future work.

Table 5.3: achieved goals.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

Our system, SLS, aims to improve the traditional learning activities by enhancing the interaction between class candidates, saving time wasted in routines, and reducing power consumption. These objectives meet the Egyptian government plans to computerize and automize its systems. However, the system can be used in all educational organization. SLS depends on the use of the smartphones and IoT technology. The main challenge we faced was to study some specialized electric fields that are related to IoT technology.

6.2 Future Work

There is a lot of work to be done to make SLS system an essential part of educational organizations. There are many technical issues we have encountered, these issues are listed below

1. Attendence info may need to be exported into excel sheet for further processing on them. This task is not complete due to lack of time and will be considered in future work.
2. Identity theft when someone registering as someone else. This issue can be solved using biometric identification (i.e. face recognition or fingerprints) to make sure that the person you are recording is the really you.
3. Another issue is to determine the exact place in the lab of a student who wants to sit anywhere not in the order of the rows. A group of sensors can be used together to determine the exact place where the person sits.
4. Our system can be extended to include many kinds of electrical appliances (such as projectors and air conditioners).

References

[1] Guide to Arduino: https://www.arduino.cc/en/Guide/Introduction

[2] Ethernet module product: https://store.arduino.com

[3] Wi-Fi module product: https://store.arduino.com

[4] Analog I/O functions: https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/

[5] Maryam Bagheri and Siavosh H. Movahed, "The Effect of the Internet of Things (IoT) on Education Business Model", Published in: 2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS). IEEE Computer Society, 435-441.

[6] Prof. Rohini Temkar, Mohanish Gupte, et al.,"Internet of Things for Smart Classrooms", IRJET Volume: 03 Issue: 07, July-2016

[7] M.R.M.Veeramanickam and Dr. M. Mohanapriya., "IOT enabled Futurus Smart Campus with effective E-Learning : i-Campus", GSTF Journal of Engineering Technology (JET) Vol.3 No.4, April 2016.

[8] Ying-Wen Bai and Yi-Te Ku, "Automatic Room Light Intensity Detection and Control Using a Microprocessor and Light Sensors", IEEE Transactions on Consumer Electronics, Vol. 54, No. 3, August 2008.

[9] J.Chandramohan , R.Nagarajan ,et al., "Intelligent Smart Home Automation and Security System Using Arduino and Wi-fi", IJESC ISSN:2319-7242, Vol. 6 Issue 3, March 2017.

[10] Archana D, Rajani B.R, et al., "Bidirectional Visitor Counter for Smart Power Management", IJSRCSEIT ISSN: 2456-3307, Vol. 4, Issue 6, May-June 2018.

[11] Gaurav Waradkar, Hitesh Ramina, et al., "Automated Room Light Controller with Visitor Counter", IJESC ISSN 2321 3361, March 2016

[12] Nabeel Salih Ali, Ali Al Farawn, et al., "Attendance and Information System using RFID and Web-Based Application for Academic Sector", IJACSA Vol. 9, No. 1, 2018

Appendix A

DVD Contents

The DVD of the project contains the following:

1. Full code of the mobile application
2. Full code for HW programming
3. Documentation of the project in PDF format
4. Presentation of the project

Please note that you’ll need to learn React, React Native, PHP, and Arduino programming language to understand the code. You can view the official documentation of each language on its website on the internet or you can watch videos if you like (i.e. YouTube tutorial). You will also need to install Android studio, Arduino IDE, WAMP sever on your machines to run the code.