CHAPTER 1

INTRODUCTION

The term Internet of Things generally refers to scenarios where network connectivity and computing capability extends to objects, sensors and everyday items not normally considered computers, allowing these devices to generate, exchange and consume data with minimal human intervention. Simply it aims on – a “hyper connected world”. It promises to offer a revolutionary, fully connected “smart” world as the relationships between objects, their environment, and people become more tightly intertwined. On the whole, the term "Internet of Things" denotes a trend where a large number of embedded devices employ communication services offered by the Internet protocols. Some of the examples of Internet of Things are August Smart Lock, Canary Smart Security System, Fitbit One – Wearables, Barcelona - Smart Cities, AT&T - Connected Car, Smart Metering, City sense - Smart street lighting, etc.

In the implementation of IoT projects, there are five main components: Sensors, Networks, Standards, Intelligent Analysis, and Intelligent Actions. In the Internet of Things (IoT) everyday objects are readable, recognizable, locatable, addressable and controllable over the Internet. To get started, you’ll need a platform for the product development team to develop and launch the product on. Some incredible popular hardware/software platform for creating interactive IoT objects and devices includes:

Raspberry Pi: A credit card sized computer with a Linux OS and capabilities for audio, video and Internet. So it's a great platform for projects requiring multimedia. It's a mini personal computer that lets software developers dive right in and start coding - no additional components needed.

Arduino: An open source prototyping platform, great at handling your hardware configuration and pushing the data to an external device for representation.

Mainly it is very easy to connect to internet and can be programmed using variety of programming languages. If the project has very limited hardware interaction but is slightly complex on the software side or need to be connected to internet, then go with Raspberry Pi. Also if we want to program using variety of programming languages (not limited to C/C++), then we should choose Raspberry Pi.

Arduino is very easy to get started. Not very powerful when compared with Raspberry Pi. If the project requires to interact with lot of complex external hardware, then we should consider using Arduino. If the project requires lot to write complex software or requires entire software stacks or protocols, then Arduino may not be the best options.

Raspberry pi is simply used to digitize our raw ideas. We have already discussed how and why raspberry pi is more efficient than Arduino. The reason why we begin to use raspberry pi is very simple. Something had changed the way kids were interacting with computers. Majority of curriculums with lessons on using Word and Excel, or writing webpages; the end of the dot-com boom; and the rise of the home PC and games console to replace the BBC Micros, Spectrum ZX and Commodore 64 machines that people of an earlier generation learned to program on. There isn’t much any small group of people can do to address problems like an inadequate school curriculum. But those students felt that they could try to do something about the situation where computers had become so expensive and arcane that programming experimentation on them had to be forbidden by parents; and to find a platform that, like those old home computers, could boot into a programming environment. Thus came the idea of creating the device which kids could buy and learn programming or hardware on – The Raspberry Pi.

HARDWARE LAYOUT

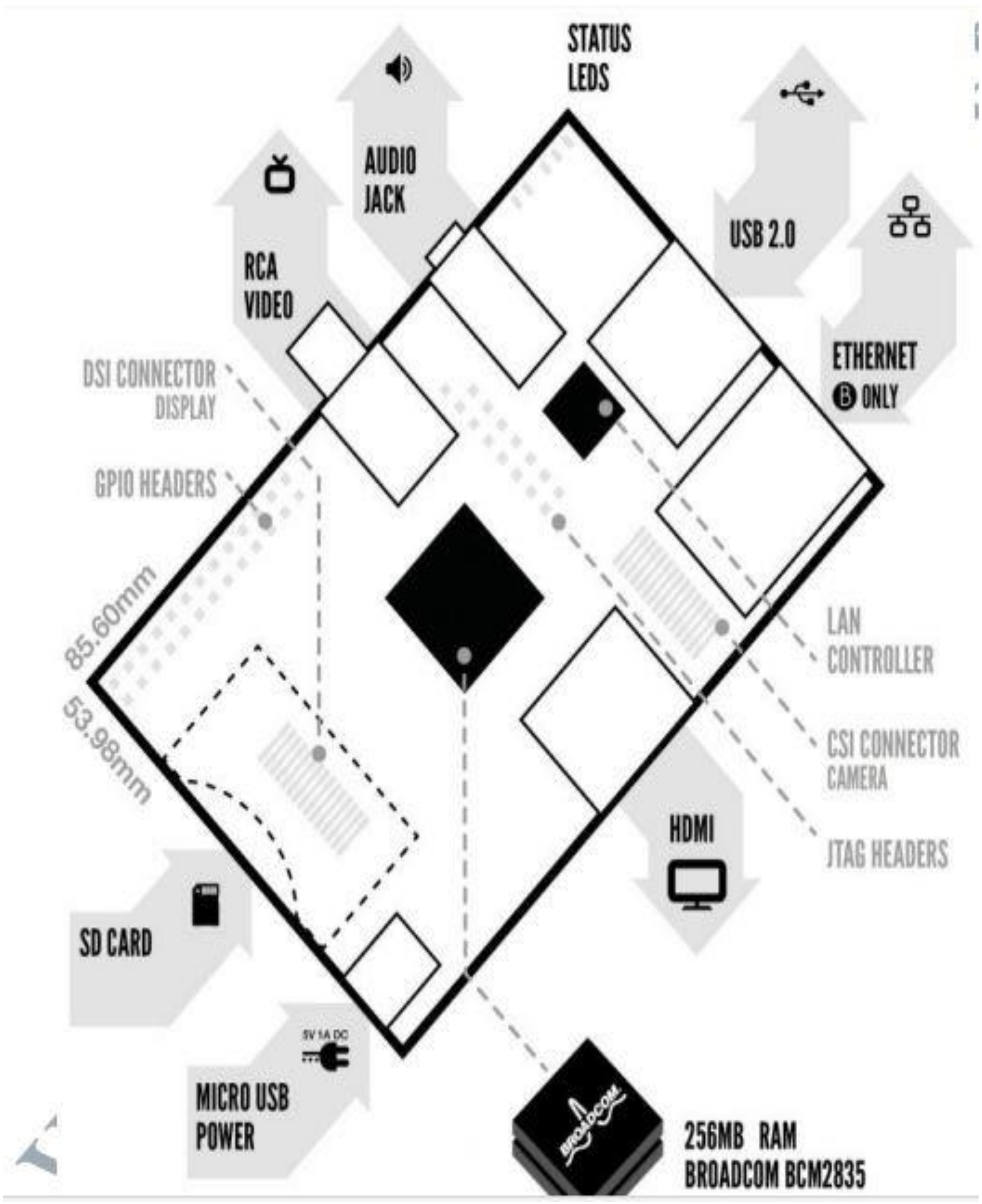


Figure 1.1 Raspberry Pi hardware layout

ARM CPU/GPU: This is a Broadcom BCM2835 System on a Chip (SoC) that's made up of an ARM central processing unit (CPU) and a Video core 4 graphics processing unit (GPU). The CPU handles all the computations that make a computer work (taking input, doing calculations and producing output), and the GPU handles graphics output.

GPIO: These are exposed general-purpose input/output connection points that will allow the real hardware hobbyists the opportunity to tinker.

RCA: An RCA jack allows connection of analog TVs and other similar output devices.

Audio out: This is a standard 3.55-millimeter jack for connection of audio output devices such as headphones or speakers. There is no audio in.

LEDs: Light-emitting diodes, for all of your indicator light needs.

USB: This is a common connection port for peripheral devices of all types (including your mouse and keyboard). Model A has one, and Model B has two. You can use a USB hub to expand the number of ports or plug your mouse into your keyboard if it has its own USB port.

HDMI: This connector allows you to hook up a high-definition television or other compatible device using an HDMI cable.

Power: This is a 5v Micro USB power connector into which you can plug your compatible power supply.

SD card slot: This is a full-sized SD card slot. An SD card with an operating system (OS) installed is required for booting the device. They are available for purchase from the manufacturers, but you can also download an OS and save it to the card yourself if you have a Linux machine and the wherewithal.

Ethernet: This connector allows for wired network access and is only available on the Model B.

1.1 PROBLEM STATEMENT

The main idea of digitalizing manual processes is to save the time spent doing things that are time consuming. This is also done to avoid human traits. Here, we are going to automate processes involved in a college or a corporate environment. The application for Leave, on-duty, gate pass and many other requisitions are done manually now days in schools and colleges. This manual system includes procedures like filling a form of requisition, getting it signed from the higher authorities and submitting it on time etc. In this process the individual has to spend time in getting the form, filling it up with required details and has to find the faculty in order to get it signed. This is really a time consuming procedure. Sometimes it is a waste of time for officials too in their hectic schedule. It’s a waste of time for both students and the faculties. This can be digitalized in order to reduce the time involved in it. Thus came the idea of smart student ID. With this device, students and faculties are connected in a local network and the request and response mechanism takes place via internet.

1.2 PROPOSED SYSTEM

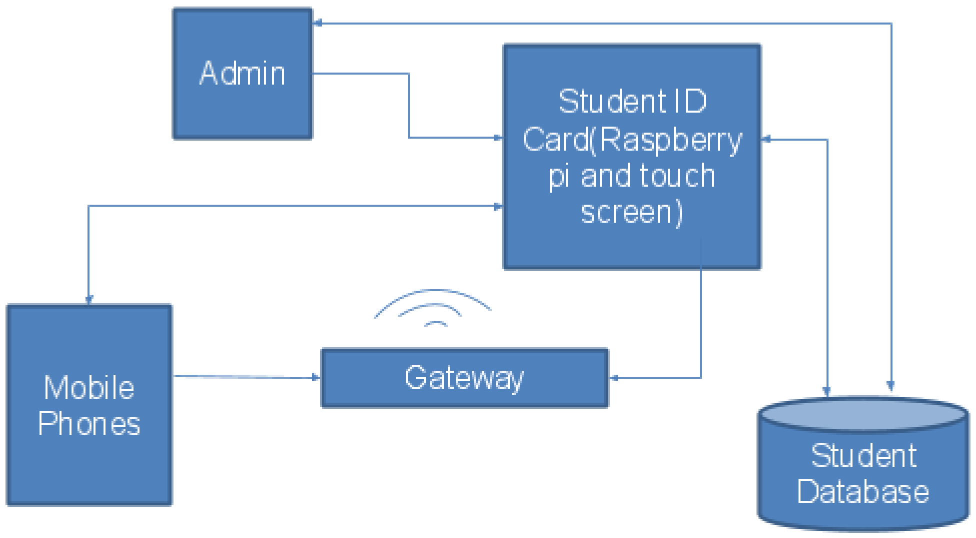


Figure 1.2 System architecture

From universities to primary schools, a revolution in technology is taking shape. Programs required an army of administrators to manage piles of paperwork is but a digital page in today's history e-books. Twenty-first-century innovation is bringing automation—and a chance to improve security—to the campus.

The above diagram represents the proposed system architecture. The existing system involves humans in processing requests. Simply the existing system is a pen and paper mechanism. This is a time consuming process. For any request to the organization, a form must be filled and must be verified by the higher authorities. The response is also in hand written format or in a printed format. In organizations like schools and even in corporate companies this traditional way is followed. We do have email technology but our idea is to integrate all in on.

The term "smart access cards" in education, business, and government applications encompasses a wide range of technologies. The common feature of most contactless, proximity-based solutions relies on embedded high-frequency (HF) RFID technology. Most contact type ID cards use magnetic stripe technology, which means students must swipe their cards through a reader, slowing down student access and creating bottlenecks. Magnetic stripe cards are also unreliable since they are easily de-magnetized. Our idea of computerizing student ID involves components like raspberry pi 3b with LCD touch screen, an android application for staffs to handle requests from students and a server running with a MYSQL database consists of student details and stores all details requests and responses via the raspberry pi. All the three are connected in a local network via Wi-Fi technology. This system includes features like,

1. Applying for leave and on-duty.
2. Receiving circulars through the device.
3. Notifying the students about their attendance percentage.
4. Automating the attendance and also a scientific calculator for calculations.

1.3 PLATFORM

We use three main platforms for the device, server and the smartphone.

1.3.1 Raspbian Jessie OS for raspberry pi 3b model

The Raspbian operating system is based on Debian Linux, and the different versions of Debian are named after characters from the “Toy Story” films. Recent versions of Raspbian have been based on Debian Wheezy (the penguin who’s lost his squeaker in “Toy Story 2”), but Raspbian has now been updated to the new stable version of Debian, which is called Jessie. The first thing anyone starting the new Jessie image from scratch will notice is that the default behaviour is to boot straight to the desktop GUI, not to the Linux command line. This was a decision taken because this is the expected behaviour for all modern computers; the default interface for a personal computer in 2015 is a desktop GUI, not just text on a screen. It is still possible to set the Pi to boot to the command line for people who prefer that. When the desktop launches, you might notice some slight tweaks to the appearance of things like menus, check boxes and radio buttons. This is because the appearance of Raspbian is now based on version 3 of GTK+, the user interface toolkit used for the LXDE desktop environment.

1.3.2 Ubuntu

Ubuntu is built on Debian’s architecture and infrastructure, to provide Linux server, desktop, phone, tablet and TV operating systems. Ubuntu releases updated versions predictably every six months, and each release receives free support for nine months (eighteen months prior to 13.04) with security fixes, high-impact bug fixes and conservative, substantially beneficial low-risk bug fixes. The first release was in October 2004.

Starting with Ubuntu 6.06, every fourth release, one release every two years, receives long-term support (LTS). Long-term support includes updates for new hardware, security patches and updates to the 'Ubuntu stack' (cloud computing infrastructure). The first LTS releases were supported for three years on the desktop and five years on the server; since Ubuntu 12.04 LTS, desktop support for LTS releases was increased to five years as well. LTS releases get regular point releases with support for new hardware and integration of all the updates published in that series to date.

Installation

Live images are the typical way for users to assess and subsequently install Ubuntu. These can be downloaded as a disk image (.iso) and subsequently burnt to a DVD and booted, or run via UNetbootin directly from a USB drive (making, respectively, a live DVD or live USB medium). Running Ubuntu in this way is typically slower than running it from a hard drive, but does not alter the computer unless specifically instructed by the user. If the user chooses to boot the live image rather than execute an installer at boot time, there is still the option to then use an installer called Ubiquity to install Ubuntu once booted into the live environment. Disk images of all current and past versions are available for download at the Ubuntu web site. Various third-party programs such as remastersys and Reconstructor are available to create customized copies of the Ubuntu Live DVDs (or CDs). "Minimal CDs" are available (for server use) that fit on a CD.

Additionally, USB flash drive installations can be used to boot Ubuntu and Kubuntu in a way that allows permanent saving of user settings and portability of the USB-installed system between physical machines (however, the computers' BIOS must support booting from USB). In newer versions of Ubuntu, the Ubuntu Live USB creator can be used to install Ubuntu on a USB drive (with or without a live CD or DVD). Creating a bootable USB drive with persistence is as simple as dragging a slider to determine how much space to reserve for persistence; for this, Ubuntu employs Casper.

1.3.3 Android

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics.

Android's source code is released by Google under an open source license, although most Android devices ultimately ship with a combination of free and open source and proprietary software, including proprietary software required for accessing Google services. Android is popular with technology companies that require a ready-made, low-cost and customizable operating system for high-tech devices. Its open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which deliver updates to older devices, add new features for advanced users or bring Android to devices originally shipped with other operating systems. The success of Android has made security an issue, in which the majority of Android devices do not receive security patches, and it has become a target for patent and copyright litigation as part of the so-called "smartphone wars" between technology companies.

1.4 TECHNOLOGIES

1.4.1 PYTHON

Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. Some programming-language features of Python are:

- A variety of basic data types are available: numbers (floating point, complex, and unlimited-length long integers), strings (both ASCII and Unicode), lists, and dictionaries.
- Python supports object-oriented programming with classes and multiple inheritance.
- Code can be grouped into modules and packages.
- The language supports raising and catching exceptions, resulting in cleaner error handling.
- Python contains advanced programming features such as generators and list comprehensions.

GUI in Python Programming

Python provides various options for developing graphical user interfaces (GUIs). The most widely used interface is tkinter.

Tkinter Programming

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps:

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by user.

Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets. Some widgets used in this project are:

Button : The Button widget is used to display buttons in your application.

Entry : The Entry widget is used to display a single-line text field for accepting values from a user.

Frame : The Frame widget is used as a container widget to organize other widgets.

Label : The Label widget is used to provide a single-line caption for other widgets. It can also contain images.

Text : The Text widget is used to display text in multiple lines.

Label Frame : A label frame is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.

tkMessageBox : This module is used to display message boxes in your applications.

We imported many python libraries for this project which handles request and response messages to and from the LAMP server running in Ubuntu 15.04.

httplib2

A comprehensive HTTP client library, httplib2 supports many features left out of other HTTP libraries. The latest version is 0.10.3.

HTTP and HTTPS: HTTPS support is only available if the socket module was compiled with SSL support.

Keep-Alive: Supports HTTP 1.1 Keep-Alive, keeping the socket open and performing multiple requests over the same connection if possible.

Authentication: The following three types of HTTP Authentication are supported. These can be used over both HTTP and HTTPS.

- Digest
- Basic
- WSSE

Caching: The module can optionally operate with a private cache that understands the Cache-Control: header and uses both the ETag and Last-Modified cache validators. Both file system and memcached based caches are supported.
All Methods: The module can handle any HTTP request method, not just GET and POST.

Redirects: Automatically follows 3XX redirects on GETs.

Compression: Handles both ‘deflate’ and ‘gzip’ types of compression.

urllib

urllib is a package that collects several modules for working with URLs. The urllib.request module defines functions and classes which help in opening URLs (mostly HTTP) in a complex world — basic and digest authentication, redirections, cookies and more.

The urllib.request module defines the following functions: urllib.request.urlopen (url, data=None, [timeout,]*, cafile=None, capath=None, cadefault=False, conte xt=None) opens the URL url, which can be either a string or a Request object. Data must be a byte object specifying additional data to be sent to the server, or none if no such data is needed. Data may also be an iterable object and in that case Content-Length value must be specified in the headers. Currently HTTP requests are the only ones that use data; the HTTP request will be a POST instead of a GET when the data parameter is provided.

The urllib.parse.urlencode() function takes a mapping or sequence of 2-tuples and returns an ASCII text string in this format. It should be encoded to bytes before being used as the data parameter. urllib.request module uses HTTP/1.1 and includes Connection:close header in its HTTP requests.

The optional timeout parameter specifies a timeout in seconds for blocking operations like the connection attempt (if not specified, the global default timeout setting will be used). This actually only works for HTTP, HTTPS and FTP connections. The optional cafile and capath parameters specify a set of trusted CA certificates for HTTPS requests. cafile should point to a single file containing a bundle of CA certificates, whereas capath should point to a directory of hashed certificate files. This function always returns an object which can work as a context manager and has methods such as

geturl(): It returns the URL of the resource retrieved, commonly used to determine if a redirect was followed. This module defines a standard interface to break Uniform Resource Locator (URL) strings up in components (addressing scheme, network location, path etc.), to combine the components back into a URL string, and to convert a "relative URL" to an absolute URL given a "base URL. The module has been designed to match the Internet RFC on Relative Uniform Resource Locators. The urllib.parse module defines functions that fall into two broad categories: URL parsing and URL quoting. These are covered in detail in the following sections.

URL Parsing

The URL parsing functions focus on splitting a URL string into its components, or on combining URL components into a URL string. urllib.parse.urlparse (urlstring, scheme='', allow_fragments=True*)* Parse a URL into six components, returning a 6-tuple. This corresponds to the general structure of a URL: scheme://netloc/path;parameters?query#fragment. Each tuple item is a string, possibly empty. The components are not broken up in smaller parts (for example, the network location is a single string), and % escapes are not expanded. The delimiters as shown above are not part of the result, except for a leading slash in the *path* component, which is retained if present.

1.4.2 LAMP STACK

LAMP is an archetypal model of web service stacks, named as an acronym of the names of its original four open-source components: the Linux operating system, the Apache HTTP Server, the MySQL relational database management system (RDBMS), and the PHP programming language. The LAMP components are largely interchangeable and not limited to the original selection. As a solution stack, LAMP is suitable for building dynamic web sites and web applications. Since its creation, the LAMP model has been adapted to other componentry, though typically consisting of free and open-source software.

Software components

Linux

Linux is a Unix-like computer operating system assembled under the model of free and open source software development and distribution. Most Linux distributions, as collections of software based around the Linux kernel and often around a package management system, provide complete LAMP setups through their packages. According to W3Techs in October 2013, 58.5% of web server market share was shared between Debian and Ubuntu, while RHEL, Fedora and CentOS together shared 37.3%.

Apache

The role of LAMP's web server has been traditionally supplied by Apache, and has since included other web servers such as Nginx. Apache has been the most popular web server on the public Internet. In June 2013, Net craft estimated that Apache served 54.2% of all active websites and 53.3% of the top servers across all domains. In June 2014, Apache was estimated to serve 52.27% of all active websites, followed by nginx with 14.36%. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Released under the Apache License, Apache is open-source software. A wide variety of features are supported, and many of them are implemented as compiled modules which extend the core functionality of Apache. These can range from server-side programming language support to authentication schemes.

MySQL and alternatives

MySQL's original role as the LAMP's relational database management system (RDBMS) has since been alternately provisioned by other RDBMSs such as Maria DB or PostgreSQL, or even NoSQL databases such as MongoDB. MySQL is a multithreaded, multi-user, SQL database management system (DBMS), acquired by Sun Microsystems in 2008, which was then acquired by Oracle Corporation in 2010. Since its early years, the MySQL team has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. Maria DB is a community-developed fork of MySQL, led by its original developers. PostgreSQL is also an ACID-compliant relational database, unrelated to MySQL. MongoDB is a widely used open-source NoSQL database that eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (calling the format BSON), making the integration of data in certain types of applications easier and faster.

PHP and alternatives

PHP's role as the LAMP's application programming language has also been performed by other languages such as Perl and Python. PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP code is interpreted by a web server via a PHP processor module, which generates the resulting web page. PHP commands can optionally be embedded directly into an HTML source document rather than calling an external file to process data. It has also evolved to include a command-line interface capability and can be used in standalone graphical applications. PHP is free software released under the terms of PHP

License, which is incompatible with the GNU General Public License (GPL) due to the restrictions PHP License places on the usage of the term PHP. Perl is a family of high-level, general-purpose, interpreted, dynamic programming languages. The languages in this family include Perl 5 and Perl 6. They provide advanced text processing facilities without the arbitrary data-length limits of many contemporary tools, facilitating manipulation of text files. Perl 5 gained widespread popularity in the late 1990s as a CGI scripting language for the Web, in part due to its parsing abilities. Python is a widely used general-purpose high-level programming language. Python supports multiple programming paradigms, including object-oriented, imperative, functional and procedural paradigms. It features a dynamic type system, automatic memory management, a standard library, and strict use of whitespace. Like other dynamic languages, Python is often used as a scripting language, but is also used in a wide range of non-scripting contexts. Specific solutions are required for websites that serve large numbers of requests, or provide services that demand high uptime. High-availability approaches for the LAMP stack may involve multiple web and database servers, combined with additional components that perform logical aggregation of resources provided by each of the servers, as well as distribution of the workload across multiple servers. The aggregation of web servers may be provided by placing a load balancer in front of them, for example by using Linux Virtual Server (LVS). For the aggregation of database servers, MySQL provides internal replication mechanisms that implement a master/slave relationship between the original database (master) and its copies (slaves).LAMP setups are capable of providing almost linear improvements in performance for services having the number of internal database read operations much higher than the number of write/update operations.

1.4.3 ANDROID

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. Applications ("apps"), which extend the functionality of devices, are written using the Android software development kit (SDK) and, often, the Java programming language. Java may be combined with C/C++, together with a choice of non-default runtimes that allow better C++ support. The SDK includes a comprehensive set of development tools, including a debugger, software libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Initially, Google's supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) plugin; in December 2014, Google released Android Studio, based on IntelliJ IDEA, as its primary IDE for Android application development. In January 2014, Google unveiled an framework based on Apache Cordova for porting Chrome HTML 5 web applications to Android, wrapped in a native application shell. Android has a growing selection of third-party applications, which can be acquired by users by downloading and installing the application's APK (Android application package) file, or by downloading them using an application store program that allows users to install, update, and remove applications from their devices. Google Play Store is the primary application store installed on Android devices that comply with Google's compatibility requirements and license the Google Mobile Services software.

CHAPTER 2

LITERATURE SURVEY

Design of Small Smart Home System by Andi Adriansyah; and Akhmad Wahyu Dani (2014)

Smart Home is applied in order to provide comfort, energy efficiency and better security. Smart Home System is still rarely used in Indonesia because of the cost and the difficulty of getting the device. The objective of this paper is to offer a Small Smart Home System designed and created by utilizing WLAN network based on Arduino microcontroller. The system is able to monitor and control lights, room temperature, alarms and other household appliances. Results this show proper control and control monitoring functions can be performed from a device connected to a network supports HTML5.

A Design of a Water Tanks Monitoring System Based on Mobile Devices by L.A. Gama-Moreno; A. Corralejo; A. Ramirez-Molina; J. A. Torres-Rangel; C. Martinez-Hernandez; and M.A. Juarez (2016)

Water supply has become a big problem over the past few years, due to overpopulation, climate change, and deterioration of pipes which can cause major problems, such as water leakage. The real problem is not (in most cases) the size of the leak, but the time it takes to detect it. In this paper, an implementation of a system to monitor the water tanks is presented. The system is called InteRface for Monitoring wAter tanks (IRMA) consists of 1) instrumentation system based on an ultrasonic sensor installed in the water tank bound to an Arduino device. This is connected to 2) the application service that receives and manages the measurements of the water tank levels. In order to advice 3) the mobile user interface over any mobile device, giving the user control everywhere and every time over the GSM network. IRMA is focused on automating the control of water tanks, such as monitoring and filling.

Surveillance and Rescue Robot using Android by Mohammad Shoeb Shah; and P. B. Borole (2016)

This paper presents an economical yet effective robotizing an Arduino microcontroller and Android Smartphone. Generally, surveillance robots consist a high cost microcontroller, video camera, GPS (Global Positioning System) module, GSM (Global System for Mobile) module, audio systems and a costly and complicated communication system. Modern Smart phones are built with the hardware that satisfy all the above requirements. The adoption of a Smartphone greatly reduces the cost of building a typical surveillance and rescue robot. The robot can be controlled remotely using Internet from a laptop or tablet. The live video feedback is obtained from the camera of Smartphone. Data from the sensors of a Smart phone such as geographic location, acceleration, etc. is sent by the Smartphone to the user. The Robot is equipped with robotic arm and various critical sensors and detectors. Features such as obstacle avoidance and solar power gives robot the freedom to explore and rescue in all type of environment.

Smartphone Application for Tracking Students' Class Attendance by Sanja Maravić Čisar; Robert Pinter; Viktor Vojnić; Vanja Tumbas; and Petar Čisar (2016)

The Bologna system makes class attendance mandatory. Recording attendance is inefficient and takes up too much class time especially with larger groups of students. If a professor records student attendance manually, they have to look at student, write it down and go to the next student. Here, the majority of students have some kind of smart device and these can be used to make an application so that students can log in their attendance. One possible solution for this is the application called Muffin. Muffin consists of a mobile application that students have on their personal mobile devices, an Arduino Uno board with Bluetooth module, and a desktop application. The basic idea is that when students want to log in their attendance, all they need to do is send some log in data to the Arduino board, which is then forwarded to the application on the desktop PC that is directly connected to the board. Tracking student attendance manually can be time consuming and error prone. The main goal of the Muffin project was to provide a fast and efficient attendance tracking system which requires minimum cost.

Multi-Functional Secured Smart Home by Shariq Suhail; Viswanatha Reddy; Rambabu; Dharma Savarni C. V.; and V. K. Mittal5 (2016)

Enhancing the home security by remote control means is a cutting-edge research area in the domain of Internet of Things (IoT). The necessity of security is increasing these days, ranging from thefts, burglary, accidents, LPG gas leakage and fire detection etc., which all are important aspects of a Home Security System. In this paper, a prototype Multi-Functional Secured Smart Home (SSH) model is developed. Generally a home security system uses signals in terms of alarm of intruder detection. However, the proposed Multi-Functional Secured Smart Home uses a mobile communication (GSM) based Home Security System, which helps to provide a better security to have systems that can be globally connected. In the proposed system a text message is sent, whenever an event from any sensor is detected, so that immediate actions could be taken by the homeowner. The proposed SSH sends SMS using GSM-Module and mail through Raspberry Pi micro-controller. The prototype SSH based smart system also uses an Arduino micro-controller board for commands processing and control. The system uses GSM technology, which provides global access to the Smart Home Security System. The prototype SSH, developed at a low cost, can be used for converting existing homes into smart homes at relatively affordable cost and with convenience. The performance testing results of the prototype SSH are encouraging.

CHAPTER 3

UML DIAGRAMS

3.1 USE CASE DIAGRAM

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

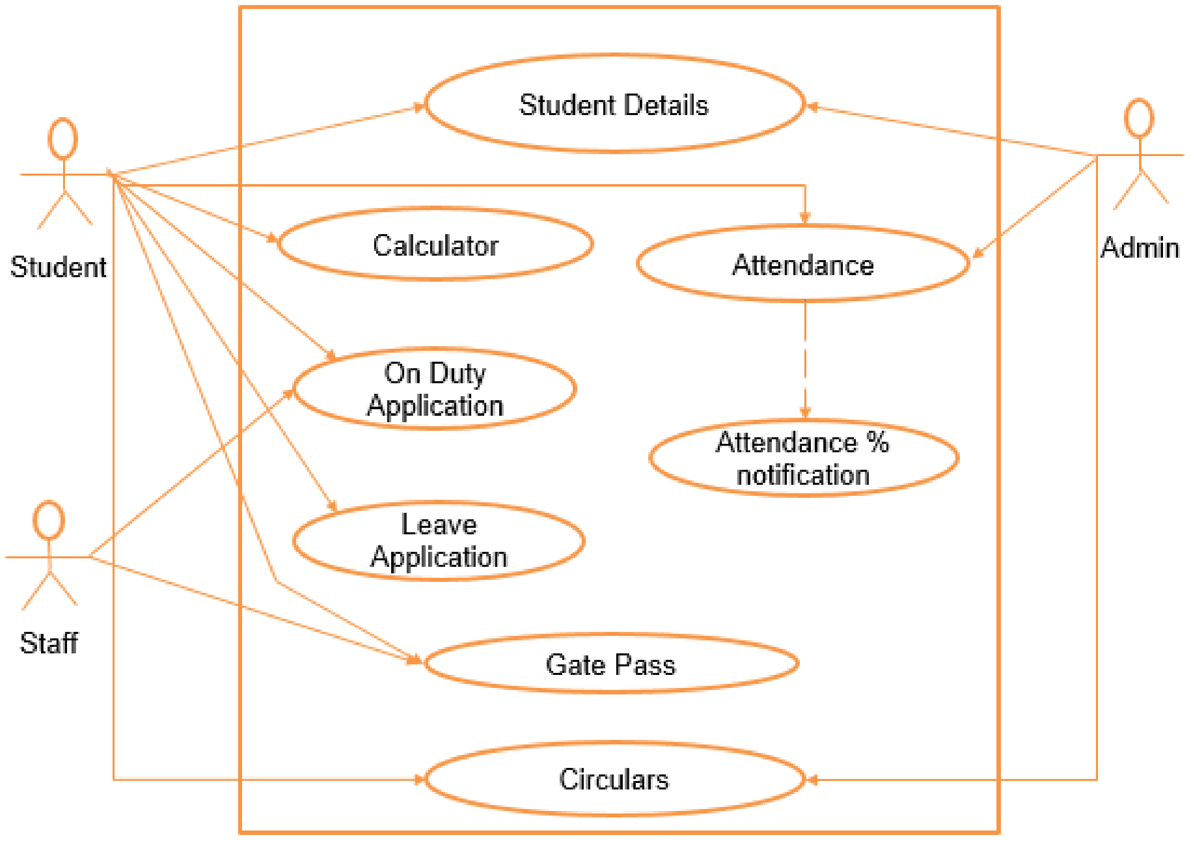


Figure 3.1 Use case diagram

This use case diagram involves three actors. Student, admin and staff. The use cases involved here are student details, calculator, on-duty application, leave application, gate pass, attendance % notification, attendance and circulars.

Admin: Admin is one of the actors involved in this use case diagram. He is the one who maintains the entire database. The functions of the admin includes modifying database in case of any new student entry. Uploading the circulars into the database for student access, flushing tables which are of no use, in an regular time interval, creating and deleting tables as per the management requirement, updating the tables if needed. Granting permissions for users and special users. Password maintenance and password changing of students if needed. Starting, restarting and stopping the server in which the database is running. Deletion of student details and tables if required, maintains the attendance details and updating the attendance percentage of each student etc.

Student: Student is one of the actors involved in this use case diagram. The role of student is to provide his/her details to the admin to be updated into the database. Student can make use of the scientific calculator application running on his device. He the one who requests for leave, on-duty and gate pass. He can view his attendance percentage and make his presence with his device by entering the code, the staff specifies. He has the access to view any circular being circulated. It keeps him informed and updated about the events happening within the campus and also regarding the college and examination fees.

Staff: Staff is one of the actors involved in this use case diagram. Every staff is associated with an android application running in their mobile phones. They are responsible to initiate the attendance session whenever attendance need to be taken. They need to respond to the students by accepting/denying the request sent by the students. Then the special staff like HOD need to approve/deny the request which is already verified by the staffs.

3.2 PROJECT ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control. Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

- Rounded rectangles represent actions;
- Diamonds represent decisions;
- Bars represent the start (split) or end (join) of concurrent activities;
- A black circle represents the start (initial node) of the workflow;
- An encircled black circle represents the end (final node).

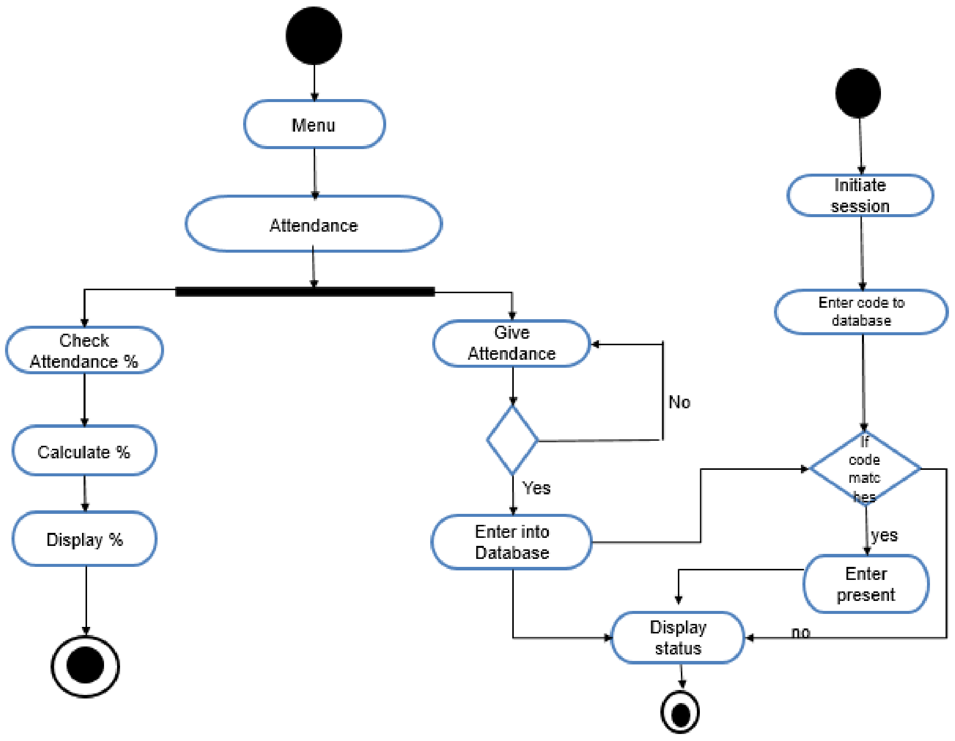


Figure 3.2 Activity diagram for attendance module

Here, let us understand the activity diagram for attendance module. First from the menu which contains all the options, attendance option is selected by the students from their smart ID. There are two options available for the students. They are:

1. Check attendance percentage
2. Give attendance

In the text box provided, they should type the code which is informed to them by the staff to mark themselves present for that day. If they choose check attendance percentage, it is calculated with the help of their data present in the database and is displayed to them on the screen. If they need give attendance they should enter the code generated in the staff application in the text box provided for them. For this the staff has to initiate the attendance session and need to read out the code which is randomly generated in their application. If the code entered by the student matched with the code which is being generated, then that particular student is marked present for that day. If, in case the code which is entered did not match with the code generated then, within 10 seconds the student has to retype the same code correctly. It is a four-digit code which includes all digits from 0 to 9 and characters a to z only lowercase. This algorithm for attendance is used to restrict the students from giving proxy. Timer is set to 10 seconds. So that the student has only limited time to enter the code and it will be difficult for him to enter code for his friend who is not in the class at the same time. This is the new concept developed by us to ensure security and reliability.

3.3 DATA FLOW DIAGRAM

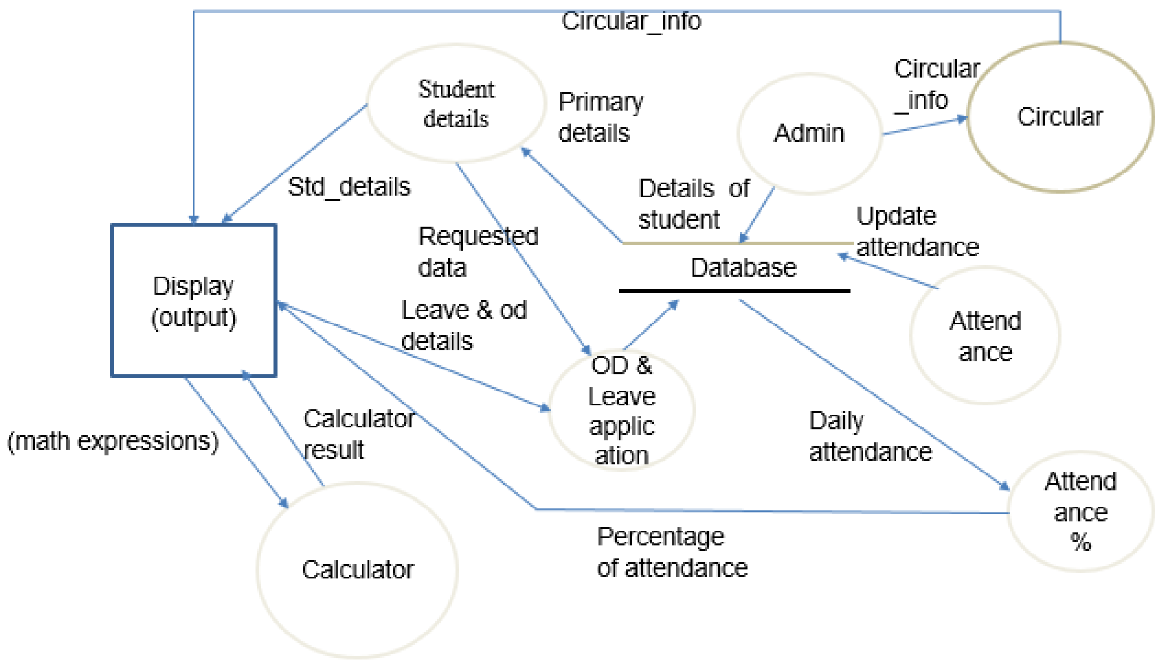


Figure 3.3 Data flow diagram

The above dataflow diagram includes functions like student details, admin, circular, attendance, attendance percentage, calculator, OD & Leave application, and a database. The functions are called whenever the corresponding button is pressed. The student details function displays the basic student details of a particular student. The attendance module enables the student to give attendance and the attendance percentage function displays the attendance percentage of that particular student. OD form and leave application module works with request and response model. Here the student fills the form he wants to apply and sends it as a request to the staff's application. This is also associated with the database where all are recorded. The staff view the requests and responds with a grant or deny reply. The Admin uploads circulars that may be viewed by students. Scientific calculator function involves a scientific calculator which is programmed in python. Students can use it whenever it is required. It solves mathematical expressions and gives the result.

CHAPTER 4

IMPLEMENTATION

4.1 PYTHON INSTALLATION IN WINDOWS 10

Visit https://www.python.org and navigate to Downloads> Windows and click Python 3.6.1 Download the 64-bit version. Wait for the Python installer to download, and then double click on it.

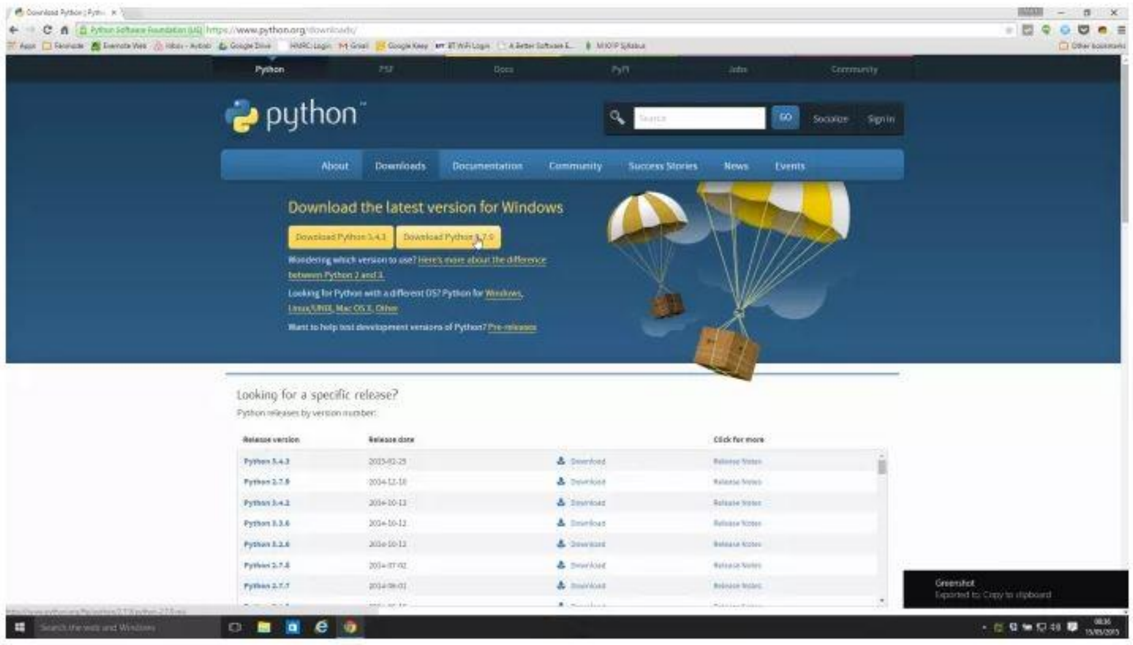

python
About
Downloads
Documentation
Community
Success Stories
News
Events
Download the latest version for Windows
Looking for a specific release?

Figure 4.1 Python download webpage

The Python Windows Installer will launch. In the Python Setup screen, select Install for all users and click Next. Give next, next, next and then finally give finish.

Now open the command prompt and go to the location where the python file is present using cd in command prompt, enter “python” into the window and hit Enter. Python has been successfully installed.

Necessary libraries are imported inside the command prompt. For example, to insert image file we need to install “tkinter” for GUI programming. In the Python command prompt, enter these command: import tkinter

To insert images

```
# pip install pill   (or)
# pip install pillow
```

Pill is the library required to insert images to the application in tkinter programming.

To handle requests and responses

```
# pip install httplib
```

A comprehensive HTTP client library, httplib2 supports many features left out of other HTTP libraries.

To work with the URLs

```
#pip install urllib
```

This module provides a high-level interface for fetching data across a network given the URL.

To import JSON

```
#pip install json
```

JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy for machines to parse and generate.

To import flask

```
#pip install flask
```

Flask is a popular, extensible web micro framework for building web applications with Python.

To perform math operation

```
#pip install math
```

In this project we imported math library to perform mathematical calculations in building the scientific calculator code.

4.2 PYTHON CODE EXPLANATION

Classes involved in the python code

class PageNav(tk.Tk)

This class involves all the functions used in page navigation. Here the frames are created and navigation from one frame to other frame is done. Containers are created here. Containers are any object that holds an arbitrary number of other objects. Generally, containers provide a way to access the contained objects and to iterate over them.

class Display(tk.Frame)

In this class the display frame is created. The display frame is the first screen which is display in the student ID. This includes, the photograph of the student and the academic details of that student. To insert the image into this frame we used PIL – library to insert images into tkinter code.

class HomePage(tk.Frame)

In this class buttons with functions are created. We included button like Student Details, Leave Application, On-Duty Application, Out Pass, Attendance Calculator and written functions to these button to perform the programmed operations. Each button has specific operation to be perform which involves request and response to and from the server.

class Student(tk.Frame)

This class includes the complete student details. It retrieves data like name, department, register number, date of birth, father name, blood group, branch etc., from the relevant table from the MySQL database running in the lamp server.

class Leave(tk.Frame)

This class involves applying leave. Here the details like name, register number of the student are retrieved from the database table and the other blanks like date of leave and reason for leave are to be filled by the student and is stored in the relevant table in the database.

class OnDuty(tk.Frame)

This class involves applying for on-duty. Here the details like name, register number of the student are retrieved from the database table and the other blanks like date of on-duty and reason for on-duty etc., are to be filled by the student and is stored in the relevant table in the database.

class OutPass(tk.Frame)

This class involves applying for out-pass. Here the details like name, register number of the student are retrieved from the database table and the other blanks like time of leaving the campus and reason are to be filled by the student and is stored in the relevant table in the database.

class Attendance(tk.Frame)

This class involves, checking attendance percentage of the student and giving attendance for the day by entering the generated code in the staff's mobile application with the given time interval.

class Calculator(tk.Frame)

This class includes the code for scientific calculator and functions involved in it. This is included in the application to make the students easy to carry a scientific calculator where ever they go and can use it whenever they require instead of carrying a physical calculator with them. All the classes except the display class contains a button "back to home" to go back to the menu page to perform the desired operation.

4.3 LAMP SERVER INSTALLATION

Installing LAMP server in Ubuntu 15.04

A "LAMP" stack is a group of open source software that is typically installed together to enable a server to host dynamic websites and web apps. We install LAMP server in Ubuntu 15.04 for database purpose. In lamp server we run MySQL database which contains tables for all the operations performed in the python code. The request and responses through the server is stored in the database. The details like reason for leave, student details etc., are stored in the database table for the management to refer in case of any issue or for any other purpose.

Steps to install LAMP server in Ubuntu 15.04

Before we begin, we should have a separate, non-root user account with “sudo” privileges set up on the server.

Step 1: Install Apache

We can install Apache easily using Ubuntu's package manager, apt. A package manager allows us to install most software pain-free from a repository maintained by Ubuntu. Use the following command:

```
sudo apt-get update
sudo apt-get install apache2
```

Since we are using a sudo command, these operations get executed with root privileges. It will ask you for your regular user's password to verify your intentions. Once we've entered your password, apt will tell us which packages it plans to install and how much extra disk space they'll take up. Press Y and hit Enter to continue, and the installation will proceed. Set Global ServerName to Suppress Syntax Warnings.

Next, we added a single line to the /etc/apache2/apache2.conf file to suppress a warning message. While harmless, if we do not set ServerName globally, we will receive the following warning when checking the Apache configuration for syntax errors:

```
sudo apache2ctl configtest
```

Open up the main configuration file with your text edit:

```
sudo nano /etc/apache2/apache2.conf
```

Type the following in the file.

```
ServerName server_domain_or_IP
```

Save and close the file when you are finished.

Restart Apache to implement your changes:

```
sudo systemctl restart apache2
```

Open the web browser and type the following:

```
http://your_server_IP_address
```

You will see the default Ubuntu 16.04 Apache web page. To Find your Server's Public IP Address:

```
ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\/.*$//'
```

Step 2: Install MySQL

MySQL is a database management system. Basically, it will organize and provide access to databases where our site can store information.

```
sudo apt-get install mysql-server
```

Again, you will be shown a list of the packages that will be installed, along with the amount of disk space they'll take up. Enter **Y** to continue.

During the installation, your server will ask you to select and confirm a password for the MySQL "root" user. This is an administrative account in MySQL that has increased privileges. Think of it as being similar to the root account for the server itself (the one you are configuring now is a MySQL-specific account, however). Make sure this is a strong, unique password, and do not leave it blank.

When the installation is complete, we want to run a simple security script that will remove some dangerous defaults and lock down access to our database system a little bit. Start the interactive script by running:

```
sudo mysql_secure_installation
```

For the rest of the questions, you should press Y and hit the Enter key at each prompt. This will remove some anonymous users and the test database, disable remote root logins, and load these new rules so that MySQL immediately respects the changes we have made.

Step 3: Install PHP

PHP is the component of our setup that will process code to display dynamic content. It can run scripts, connect to our MySQL databases to get information, and hand the processed content over to our web server to display. We're going to include some helper packages as well, so that PHP code can run under the Apache server and talk to our MySQL database:

```
sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql
```

We modify the way that Apache serves files when a directory is requested. Currently, if a user requests a directory from the server, Apache will first look for a file called index.html. We want to tell our web server to prefer PHP files, so we'll make Apache look for an index.php file first.

To do this, type this command to open the dir.conf file in a text editor with root privileges:

```
sudo nano /etc/apache2/mods-enabled/dir.conf
<IfModule mod_dir.c>
        DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm
</IfModule>
```

We want to move the PHP index file above to the first position after the DirectoryIndex specification, like this:

```
<IfModule mod_dir.c>
        DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm
</IfModule>
```

When you are finished, save and close the file by pressing Ctrl-X**.** You'll have to confirm the save by typing Y and then hit Enter to confirm the file save location.

After this, we need to restart the Apache web server in order for our changes to be recognized. You can do this by typing this:

```
sudo systemctl restart apache2
```

We can also check on the status of the apache2 service using systemctl:

```
sudo systemctl status apache2
```

Install PHP Modules

To enhance the functionality of PHP, we can optionally install some additional modules. To see the available options for PHP modules and libraries, you can pipe the results of apt-cache search into less, a pager which lets you scroll through the output of other commands:

```
apt-cache search php- | less
```

Use the arrow keys to scroll up and down, and q to quit. The results are all optional components that you can install. It will give you a short description for each. To install php module type the following command:

```
apt-get install php
```

Step 4: Test PHP Processing on your Web Server

In order to test that our system is configured properly for PHP, we can create a very basic PHP script called info.php. In order for Apache to find the file and serve it correctly, it must be saved to a very specific directory, which is called the "web root".

```
sudo nano /var/www/html/info.php
```

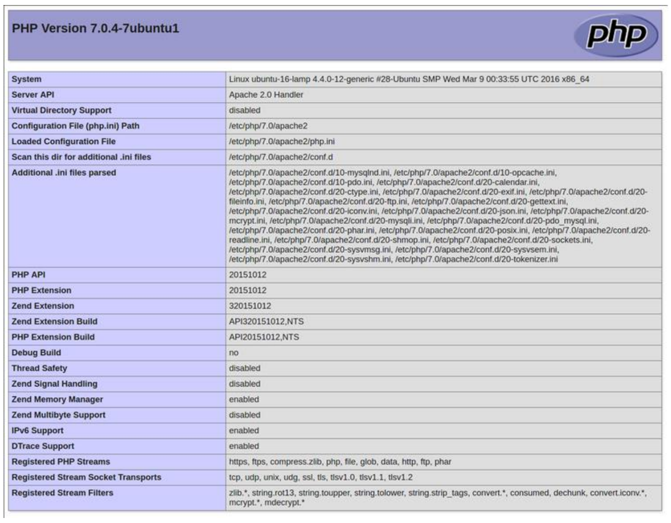
This will open a blank file. We want to put the following text, which is valid PHP code, inside the file and then, save and close it.

```
<?php
phpinfo();
?>
```

Now we can test whether our web server can correctly display content generated by a PHP script by visiting this page in our web browser. The address you want to visit will be:

```
http://your_server_IP_address/info.php
```

The page that you come to should look something like this:



PHP Version 7.0.4-7ubuntu1

System	Linux ubuntu-16-lamp 4.4.0-12-generic #28-Ubuntu SMP Wed Mar 9 00:33:55 UTC 2016 x86_64
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-mcrypt.ini, /etc/php/7.0/apache2/conf.d/20-mysqli.ini, /etc/php/7.0/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*, mcrypt.*, mdecrypt.*

Figure 4.2 Displaying info.php

4.4 ANDROID APPLICATION

Android application development

Our Android application is built using Android Studio 2.2.3 for API 21 Android Version 5.0 - Lollipop.

Volley

Volley is an HTTP library that makes networking for Android apps easier and most importantly, faster. Volley offers the following benefits:

- Automatic scheduling of network requests.
- Multiple concurrent network connections.
- Transparent disk and memory response caching with standard HTTP cache coherence.
- Support for request prioritization.
- Cancellation request API. You can cancel a single request, or you can set blocks or scopes of requests to cancel.
- Ease of customization, for example, for retry and back off.
- Strong ordering that makes it easy to correctly populate your UI with data fetched asynchronously from the network.
- Debugging and tracing tools.

The core Volley library is developed on GitHub and contains the main request dispatch pipeline as well as a set of commonly applicable utilities, available in the Volley "toolbox."

The easiest way to add Volley to a project is to add the dependency to the app's build.gradle file.

```
dependencies{

        …

        compile ‘com.android.volley:volley:1.0.0’

}
```

Add the INTERNET Permission

To use Volley, you must add the android.permission.INTERNET permission to your app's manifest. Without this, your app won't be able to connect to the network.

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Volley.newRequestQueue

Volley provides a convenience method Volley.newRequestQueue that sets up a RequestQueue for you, using default values, and starts the queue.

```
  StringRequest stringRequest = new
StringRequest(Request.Method.POST, REGISTER_URL,
    new Response.Listener<String>()
   {
       @Override
       public void onResponse(String response)
       {
           String check = response.trim();
           if (check.equals("success"))
           {
               Intent i =  new Intent(getApplicationContext(),
MainActivity2.class);
               i.putExtra("Uname",username);
```

```
                startActivity(i);
              }
             else
             {
                startActivity...
```

```
                    RequestQueue requestQueue = Volley.newRequestQueue(this);
                    requestQueue.add(stringRequest)
```

In the login activity, the user name and passwords are validated.

```
          public static final String KEY_NAME = "username";
          public static final String KEY_REGNO = "pwd";


         public void onResponse(String response) {
                    String check = response.trim();
                    if (check.equals("success")){
                       Intent i =  new Intent(getApplicationContext(),
MainActivity2.class);
                       i.putExtra("Uname",username);
                       startActivity(i);
                    }
                    else {
                       Toast.makeText(MainActivity.this, response,
Toast.LENGTH_LONG).show();
         }
```

JsonArrayRequest

```
         public JsonArrayRequest(String url,
                         Response.Listener<JSONArray> listener,
                         Response.ErrorListener errorListener)
```

Creates a new request.

It requires the following parameters:

url - URL to fetch the JSON from

listener - Listener to receive the JSON response

errorListener - Error listener, or null to ignore errors.

Python installation in raspberry pi

Updating system

```
$ sudo apt-get update
$ sudo apt-get -y upgrade
$ sudo apt-get dist-upgrade
```

Installing Python manually

```
$ sudo apt-get -y install libbz2-dev liblzma-dev libsqlite3-dev libncurses5-dev libgdbm-dev zlib1g-dev libreadline-dev libssl-dev tk-dev build-essential libncursesw5-dev libc6-dev openssl

$ cd ~

$ wget https://www.python.org/ftp/python/3.6.0/Python-3.6.0.tgz

$ tar -zxvf Python-3.6.0.tgz

$ cd Python-3.6.0

$ ./configure

$ make

$ sudo make instal
```

Verifying installation

```
python3.6 –version
```

Installing pip

```
$ cd ~

$ wget https://bootstrap.pypa.io/get-pip.py

$ sudo python3.6 get-pip.py
```

Testing installed pip

```
$ pip3.6 –version
```

CHAPTER 5

PERFORMANCE METRICS

The connectivity of the Raspberry Pi to the server installed on a computer in the laboratory or that of the Android application to the server depend on the strength of the Wi-Fi signal. The Wi-Fi network is connected to the wired network to which the server is connected. The access time to the database and the updates to the database upon interactions are instantaneous. The Pi 2 had a quad-core ARM Cortex-A7 chip, but the Pi 3's chip has been upgraded to a newer Cortex-A53. Compared to the Raspberry Pi 2 it has a 1.2GHz 64-bit quad-core ARMv8 CPU, 802.11n Wireless LAN, Bluetooth 4.1 and Bluetooth Low Energy (BLE). The Raspberry Pi 3 has a tiny Wi-Fi antenna which at 10m distance from the router and through a couple of walls, managed to transfer data at 12Mbit/s, compared to 26Mbit/s for an 802.11n laptop. The Pi 3's chip was running at 1.2GHz during the benchmark, which is 300MHz quicker than the Pi 2's chip. The built-in Wi-Fi provides capable performance and is well suited for using the device as a student ID.

CHAPTER 6
CONCLUSION AND FUTURE WORK

The digitization of our world has been a forward march for years now; still, it might be surprising to step back and look at how physical copies of our stuff have changed into something else entirely. And even more surprising might be to go beyond the debate of the level of pleasantness of reading a paperback book versus an e-book on a Kindle and look at how digitizing everything might save space and shrink the environmental footprint of everything we produce, but also put the longevity of our information at risk. The idea behind our project to digitize all our paper stuffs. Digitizing information makes it easier to preserve, access, and share. Our aim is to make work easier, make documents easy to share, storing information for a long term and in a secured manner, easily accessible by all etc., By using our model for computerizing student ID many of the paper works will get reduced, time consumed will be less, very efficient and will be error free. The idea behind marking attendance is new to the world and is very much efficient other than marking in paper and uploading it later. Though we included many features in our model, there are still many features to be included for becoming smarter than ever. May be in the future many more concepts and features will be included in our current model. Some of our future work includes:

A. **REDUCTION IN SIZE:** Our current model size is quite big to be an access card. Our future work will be on reducing the size of our current model. In this growing digital world, the size of raspberry pi will be reduced using Nano technology. There will be no issue with the size in the future.
B. **INTRODUCING BATTERY:** Our current model runs with the help of a power cable connected to a computer. To issue to each and every one in an organization, it should run on a battery to use it for the whole day.

Later it can be charged by plugging in with the computer or a laptop. Our model is a prime model. Battery concept will be one of the major future work to be done as earlier as possible.

C. **INTRODUCING FINGER PRINT:** We are planning to introduce finger print technology to our model for ensuring more security. More security is needed because it will contain all the details and history of that particular person. A hardware need to be introduced to enable this functionality, but we are planning to build a software for this purpose.
D. **UPLOADING CIRCULARS AS IMAGE FILES:** We are planning to upload the circulars by capturing it in the mobile camera which uses the staff application. The staffs can take a picture of the circular and can send directly to any particular group of student or even any particular department, instead for putting it on the notice board.

These are the future work we are planning to do. Many other features will be added in the near future for providing more efficiency and reliability. We ensure that we will improvise our model in all aspects and will produce the best product to make changes in the paper and pen concept which is widely used.

APPENDIX

APPENDIX 1

1.1 Source Code

1.1.1. Building User Interface in Python Using tkinter Programming

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import tkinter as tk
import math
import httplib2
import urllib
import flask
import json
from tkinter import *
from PIL import Image, ImageTk
LARGE_FONT = ('Verdana', 12)
##def onLeftDrag(event):
##    a = PageNav();
##    controller.show_frame(HomePage);

class PageNav(tk.Tk):

    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)
        container = tk.Frame(self)
        container.pack(side='top', fill='both', expand=True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)
    self.frames = {}
    # Insert all the frames here as class files
    for F in (
        Display,
        HomePage,
        Student,
        Leave,
        OnDuty,
        OutPass,
        Attendance,
        Calculator,
        ):
        frame = F(container, self)
        self.frames[F] = frame
        frame.grid(row=0, column=0, sticky='nsew')
    self.show_frame(Display)

 def show_frame(self, cont):
    frame = self.frames[cont]
    frame.tkraise()
class Display(tk.Frame):
   def __init__(self, parent, controller):
    tk.Frame.__init__(self, parent)
    self.configure(background='light blue')
    make_frame = LabelFrame(self, text="", width=100, height=100)
    #make_frame.bind("<Button-1>", onLeftDrag)
    make_frame.pack()
    # create the PIL image object:
    PIL_image = Image.open("Images\logoname.png")
```

```
      width = 1400
      height = 180
     # You may prefer to use Image.thumbnail instead
      # Set use_resize to False to use Image.thumbnail
      use_resize = True
if use_resize:
          # Image.resize returns a new PIL.Image of the specified size
         PIL_image_small=PIL_image.resize((width,height),Image.ANTIALIAS)
      else:
          # Image.thumbnail converts the image to a thumbnail, in place
          PIL_image_small = PIL_image
          PIL_image_small.thumbnail((width,height), Image.ANTIALIAS)

      # now create the ImageTk PhotoImage:
      img = ImageTk.PhotoImage(PIL_image_small)
      in_frame = Label(make_frame, image = img)
      #in_frame.bind("<Button-1>", onLeftDrag)
      in_frame.image= img
      in_frame.place(x=0, y=0)
      in_frame.pack(expand=YES, fill=X)
      make_frame1 = LabelFrame(self, text="", width=140, height=160)
      #make_frame1.bind("<Button-1>", onLeftDrag)
      make_frame1.pack(side=LEFT)



      # create the PIL image object:
      PIL_image1 = Image.open("Images\depika.jpg")

      width = 400
```

```
        height = 600

        # You may prefer to use Image.thumbnail instead
        # Set use_resize to False to use Image.thumbnail
        use_resize = True

        if use_resize:
            # Image.resize returns a new PIL.Image of the specified size
            PIL_image_small1 = PIL_image1.resize((width,height),
Image.ANTIALIAS)
        else:
            # Image.thumbnail converts the image to a thumbnail, in place
            PIL_image_small1 = PIL_image1
            PIL_image_small1.thumbnail((width,height), Image.ANTIALIAS)

        # now create the ImageTk PhotoImage:
        img1 = ImageTk.PhotoImage(PIL_image_small1)
        in_frame1 = Label(make_frame1, image = img1)
        #in_frame1.bind("<Button-1>", onLeftDrag)
        in_frame1.image= img1
        in_frame1.place(x=0, y=0)
        in_frame1.pack(side=LEFT, expand=YES, fill=BOTH)

        make_frame2 = LabelFrame(self, text="", width=200, height=160,
background= 'light grey')

        #make_frame2.bind("<Button-1>", onLeftDrag)
        make_frame2.pack(side=BOTTOM)
        var = StringVar()
        label = Label( self, textvariable=var, width=500, height=300)
        #label.bind("<Button-1>", onLeftDrag)
        label.config(font=("Lucida Bright", 40))
        label.pack(side=LEFT)
        in_frame2 = Label(make_frame2, text = var.set("NAME :  R.M.DEPIKA
\nDEPT  :  CSE \nDURATION  :  2013-2017\nROLL NO :  13CS009"), fg =
'light blue')
        in_frame2.configure(bg='light blue')
        buttonHome = tk.Button(make_frame2, text='Menu',
                            command=lambda : \
                            controller.show_frame(HomePage), anchor = "s")
        buttonHome.pack(side = 'bottom', expand = True)
        buttonHome.config(font=("Lucida Bright", 25))
        #in_frame2.bind("<Button-1>", onLeftDrag)
        in_frame2.pack(side=LEFT)


class HomePage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.configure(background='light blue')
        label = tk.Label(self, text='Start Page', font=LARGE_FONT, bg = 'light
blue')
        label.pack(pady=10, padx=10)
        button_label = [
            'Student Details',
            'Leave Application',
            'On-Duty Application',
           'Out Pass',
           'Attendance',
           'Calculator',
           ]

        button1 = tk.Button(self, text=button_label[0],
                    command=lambda : \
                      controller.show_frame(Student), width=65, height=2,
anchor = "center", bg = '#ffcccc')
        button1.config(font=("Lucida Bright", 25))
        button1.pack()

        button2 = tk.Button(self, text=button_label[1],
                    command=lambda : \
                    controller.show_frame(Leave), width=65, height=2, anchor =
"center", bg = '#ff8533')
        button2.config(font=("Lucida Bright", 25))
        button2.pack()

        button3 = tk.Button(self, text=button_label[2],
                    command=lambda : \
                    controller.show_frame(OnDuty), width=65, height=2, anchor
= "center", bg = '#e6b3ff')
        button3.config(font=("Lucida Bright", 25))
        button3.pack()

        button4 = tk.Button(self, text=button_label[3],
                    command=lambda : \
                    controller.show_frame(OutPass),  width=65,  height=2,  anchor
```

```
= "center", bg = '#ffd9b3')
        button4.config(font=("Lucida Bright", 25))
        button4.pack()

        button5 = tk.Button(self, text=button_label[4],
                            command=lambda : \
                            controller.show_frame(Attendance),    width=65,    height=2,
anchor = "center", bg = '#88cc00')
        button5.config(font=("Lucida Bright", 25))
        button5.pack()

        button6 = tk.Button(self, text=button_label[5],
                            command=lambda : \
                            controller.show_frame(Calculator),    width=65,    height=2,
anchor = "center", bg = '#e6e600')
        button6.config(font=("Lucida Bright", 25))
        button6.pack()
        buttonHome = tk.Button(self, text='Back to Home',
                           command=lambda : \
                           controller.show_frame(Display))
        buttonHome.pack()
        buttonHome.config(font=("Lucida Bright", 25))
```

1.1.2. PHP code

Student Records

```
<?php
#$pname=$_POST['name'];
$pregno=$_POST['regno'];
$con = mysql_connect("localhost","root","dhl1925702");
```

```
if (!$con)
{
	die('Could not connect: ' . mysql_error());
}
mysql_select_db("collegedb", $con);
$result = mysql_query("SELECT
Sname,Sregno,Sdept,Sbatch,Scont,Sfname,Sdob,Sbldgp,Scont,Saddr FROM
student_details WHERE Sregno = '".$pregno."';");
if(result)
{}
while($row = mysql_fetch_assoc($result))
{
	$output[]=$row;
}
print(json_encode($output));
mysql_close($con);
?>
```

1.1.3. Android Application Code for Login

MainActivity.java

```
package com.example.hannah.staffapp;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import java.util.HashMap;
import java.util.Map;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener {

    private static final String REGISTER_URL =
"http://192.168.3.70/loginMobile.php";

    public static final String KEY_NAME = "username";
    public static final String KEY_REGNO = "pwd";



    private EditText editTextUsername;
    private EditText editTextPwd;

    private Button buttonRegister;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        editTextUsername = (EditText) findViewById(R.id.editText);
        editTextPwd = (EditText) findViewById(R.id.editText1);

        buttonRegister = (Button) findViewById(R.id.button);

        buttonRegister.setOnClickListener(this);
    }

    private void registerUser(){
        final String username = editTextUsername.getText().toString().trim();
        final String pwd = editTextPwd.getText().toString().trim();

        StringRequest stringRequest = new StringRequest(Request.Method.POST, 
REGISTER_URL,
                new Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        String check = response.trim();
                        if (check.equals("success")){
                            Intent i =  new Intent(getApplicationContext(), 
MainActivity2.class);
                            i.putExtra("Uname",username);
                            startActivity(i);
                        }
                        else {
                            Toast.makeText(MainActivity.this, response, 
Toast.LENGTH_LONG).show();
```

```
                    }
                }
            },
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {

Toast.makeText(MainActivity.this,error.toString(),Toast.LENGTH_LONG).sho
w();
                    Log.e("Hannah testing", error.toString());
                }
            }){
        @Override
        protected Map<String,String> getParams(){
            Map<String,String> params = new HashMap<String, String>();
            params.put(KEY_NAME,username);
            params.put(KEY_REGNO,pwd);
            return params;
        }

    };

    RequestQueue requestQueue = Volley.newRequestQueue(this);
    requestQueue.add(stringRequest);
}

@Override
public void onClick(View v) {
    if(v == buttonRegister){
        registerUser();
    }
}
}
```

activity_main.xml

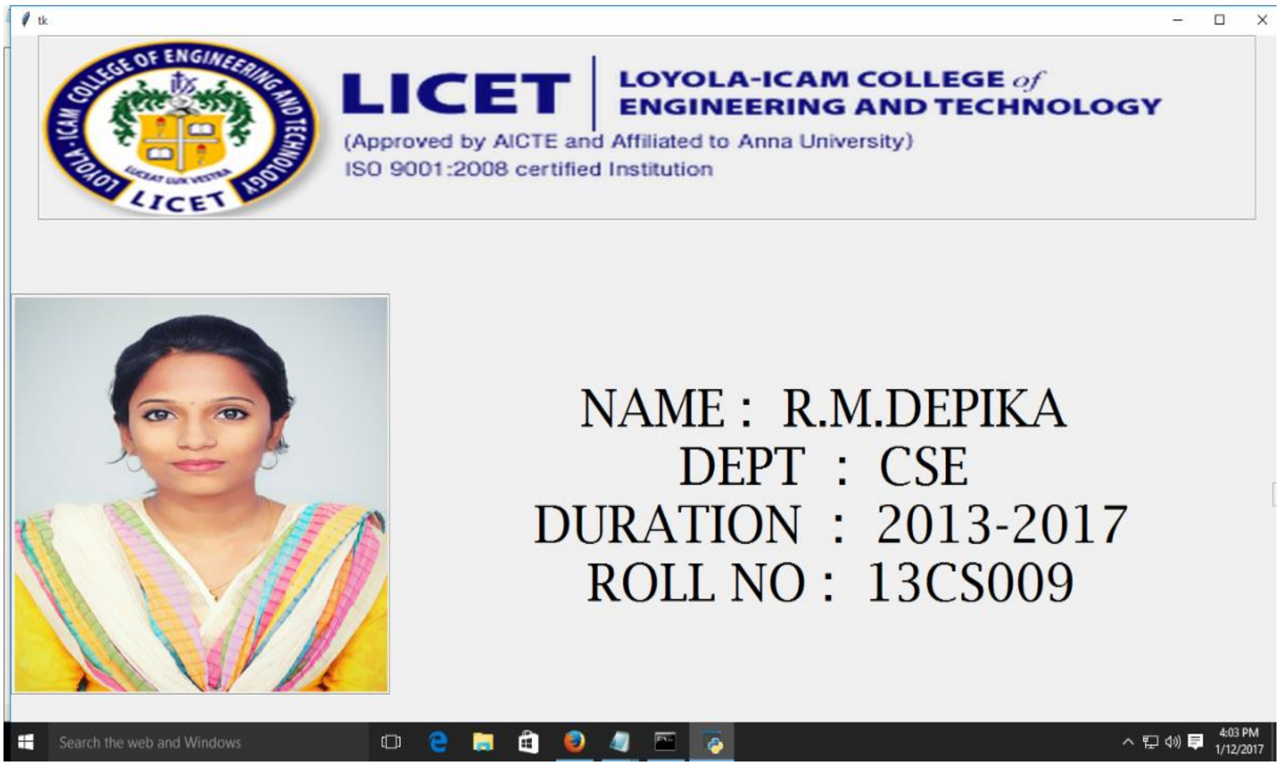
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="@android:color/holo_blue_light"
    tools:context="com.example.hannah.staffapp.MainActivity">

    <Button
        android:text="Login"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="90dp" />
```

```
<TextView
    android:text="Password"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textView2"
    android:background="@android:color/holo_blue_light"
    android:textSize="24sp"
    android:layout_above="@+id/button"
    android:layout_toStartOf="@+id/button"
    android:layout_marginEnd="12dp"
    android:layout_marginBottom="73dp" />

<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textPassword"
    android:ems="10"
    android:background="@android:color/white"
    android:id="@+id/editText1"
    android:layout_alignBottom="@+id/textView2"
    android:layout_alignStart="@+id/button" />

<TextView
    android:text="Username"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textView"
    android:background="@android:color/holo_blue_light"
    android:textSize="24sp"
```

```
        android:layout_above="@+id/textView2"
        android:layout_alignStart="@+id/textView2" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:background="@android:color/white"
        android:id="@+id/editText"
        android:layout_alignBottom="@+id/textView"
        android:layout_toEndOf="@+id/textView"
        android:layout_alignStart="@+id/editText1" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="LICET Staff App"
        android:id="@+id/textView1"
        android:background="@android:color/holo_blue_light"
        android:fontFamily="monospace"
        android:textSize="24sp"
        android:layout_marginEnd="49dp"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_marginTop="26dp" />

</RelativeLayout>
```

APPENDIX 2

2.1 Screenshots – Student ID User Interface


tk

LICET | LOYOLA-ICAM COLLEGE of ENGINEERING AND TECHNOLOGY

(Approved by AICTE and Affiliated to Anna University)

ISO 9001:2008 certified Institution

NAME : R.M.DEPIKA

DEPT : CSE

DURATION : 2013-2017

ROLL NO : 13CS009

Figure 7.1 Student ID display

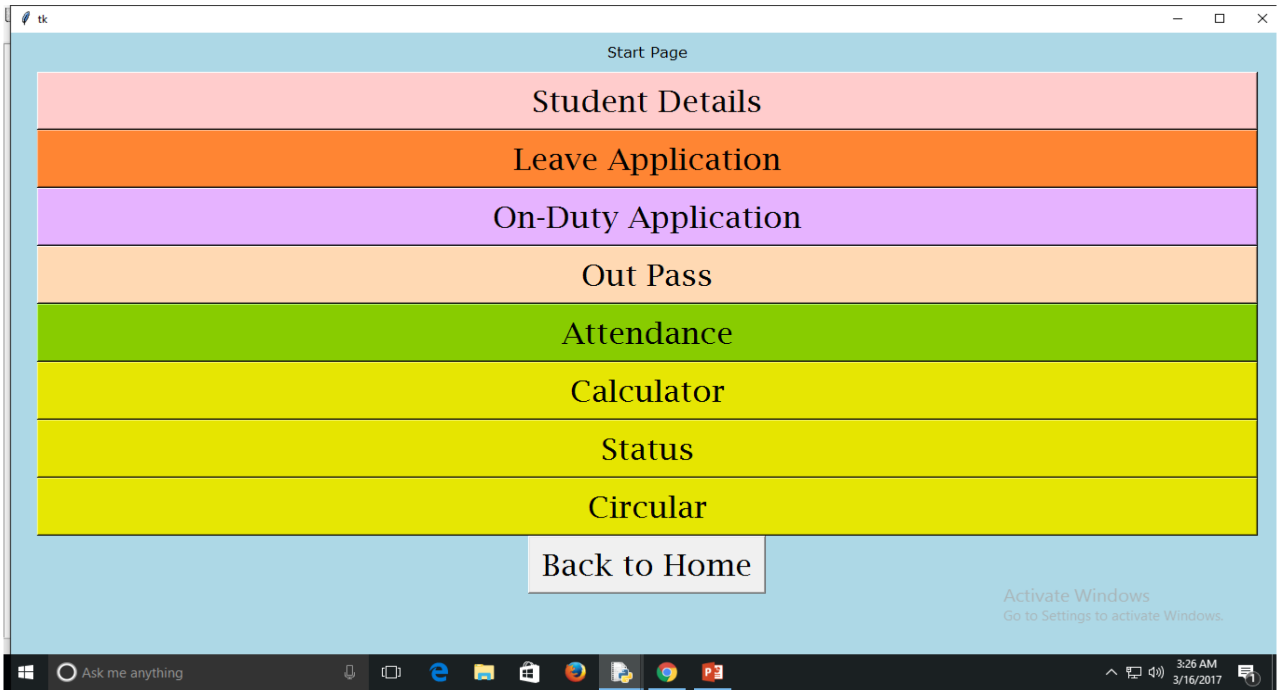
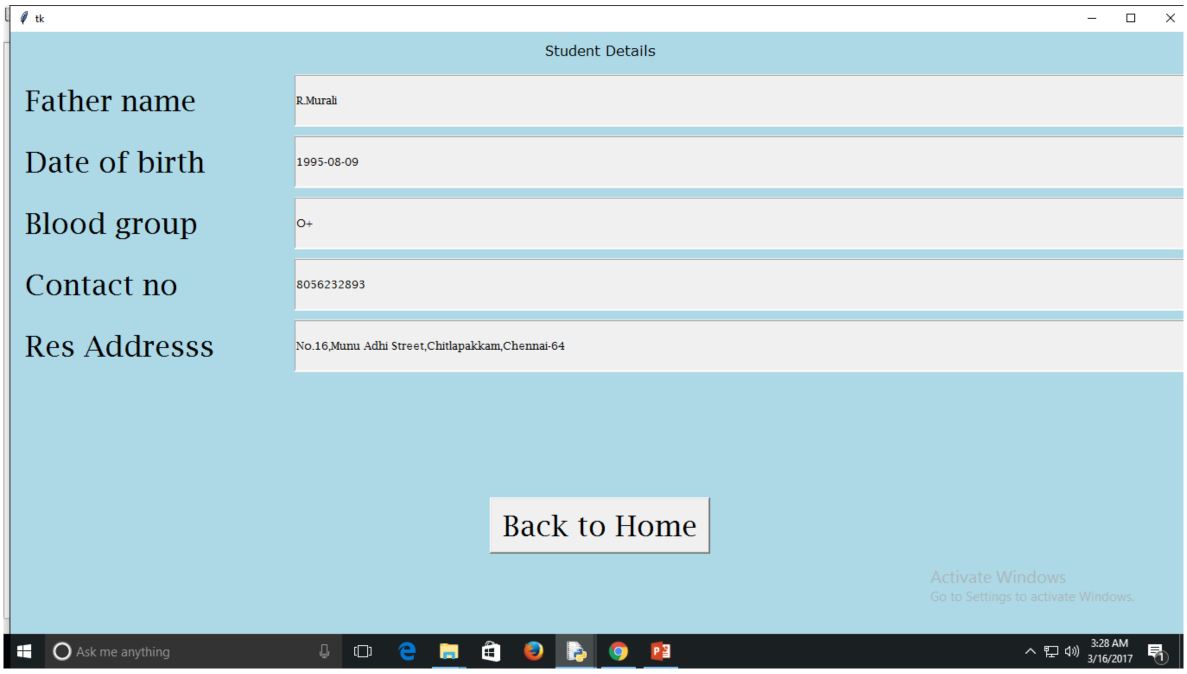


Figure 7.2 Home page display


Student Details

Father name: R.Murali
Date of birth: 1995-08-09
Blood group: O+
Contact no: 8056232893
Res Addresss: No.16,Munu Adhi Street,Chitlapakkam,Chennai-64

Figure 7.3 Student details display

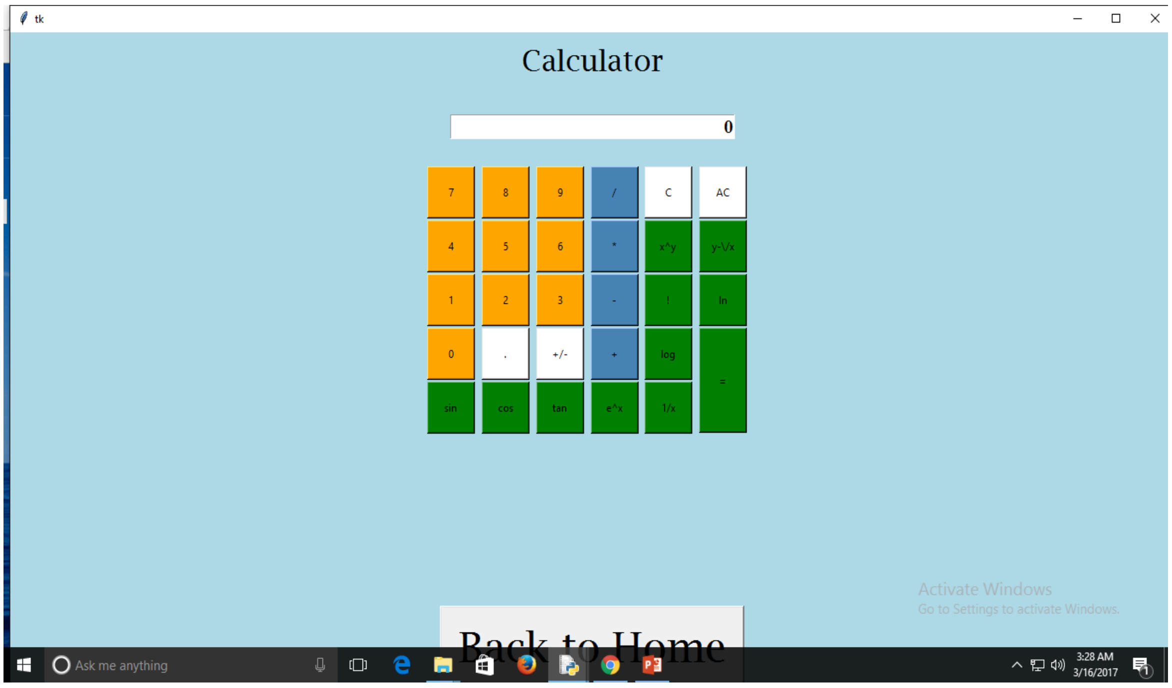
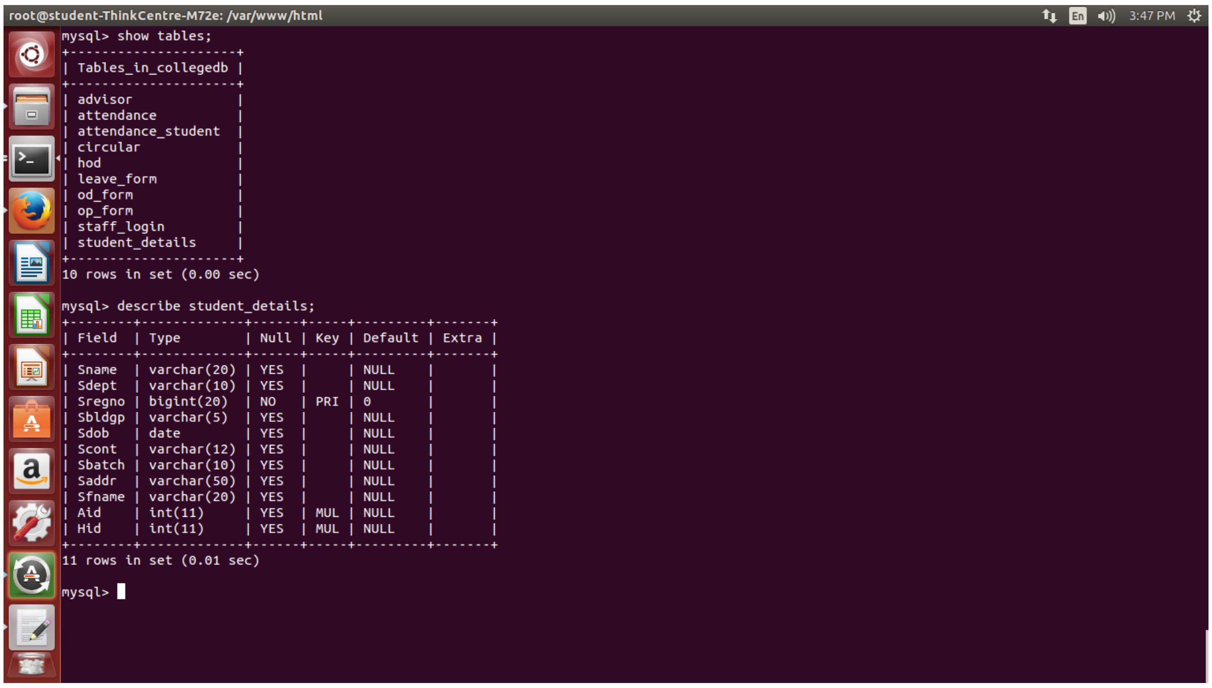


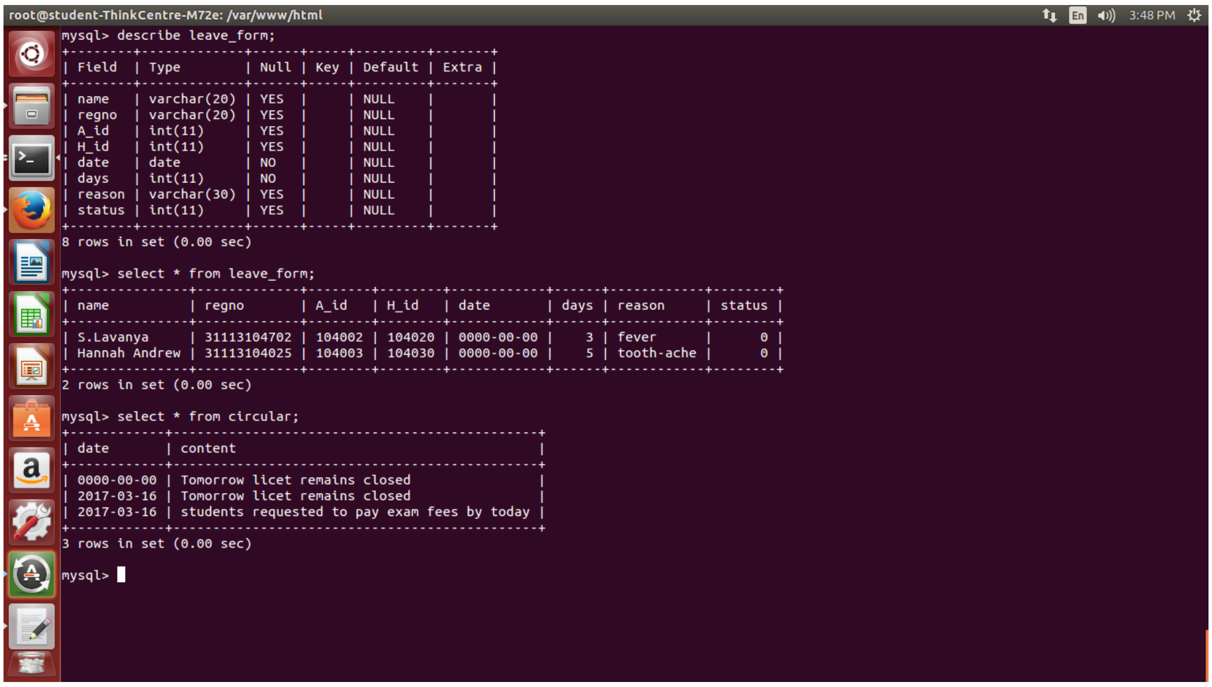
Figure 7.4 Scientific calculator display

2.2 Screenshots – Database



```
mysql> show tables;
+--------------------+
| Tables_in_collegedb |
+--------------------+
| advisor            |
| attendance         |
| attendance_student |
| circular           |
| hod                |
| leave_form         |
| od_form            |
| op_form            |
| staff_login        |
| student_details    |
+--------------------+
10 rows in set (0.00 sec)

mysql> describe student_details;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| Sname  | varchar(20) | YES  |     | NULL    |       |
| Sdept  | varchar(10) | YES  |     | NULL    |       |
| Sregno | bigint(20)  | NO   | PRI | 0       |       |
| Sbldgp | varchar(5)  | YES  |     | NULL    |       |
| Sdob   | date        | YES  |     | NULL    |       |
| Scont  | varchar(12) | YES  |     | NULL    |       |
| Sbatch | varchar(10) | YES  |     | NULL    |       |
| Saddr  | varchar(50) | YES  |     | NULL    |       |
| Sfname | varchar(20) | YES  |     | NULL    |       |
| Aid    | int(11)     | YES  | MUL | NULL    |       |
| Hid    | int(11)     | YES  | MUL | NULL    |       |
+--------+-------------+------+-----+---------+-------+
11 rows in set (0.01 sec)

mysql> █
```

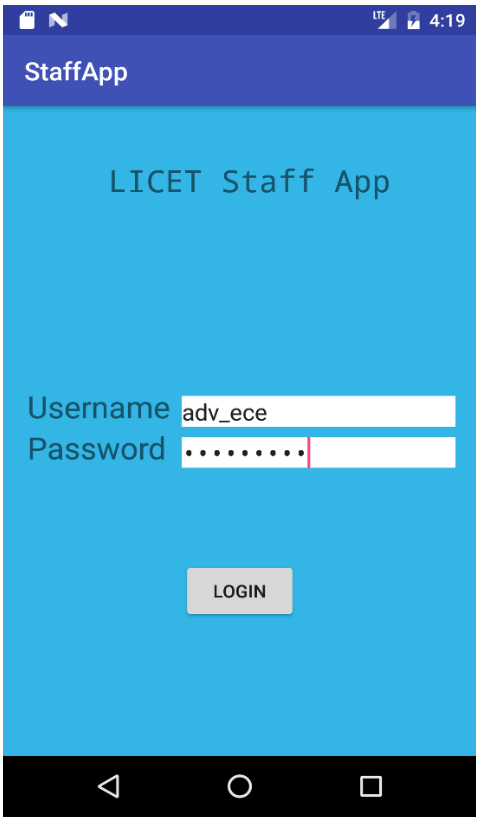
Figure 7.5 Description of student details table



```
mysql> describe leave_form;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| name   | varchar(20) | YES  |     | NULL    |       |
| regno  | varchar(20) | YES  |     | NULL    |       |
| A_id   | int(11)     | YES  |     | NULL    |       |
| H_id   | int(11)     | YES  |     | NULL    |       |
| date   | date        | NO   |     | NULL    |       |
| days   | int(11)     | NO   |     | NULL    |       |
| reason | varchar(30) | YES  |     | NULL    |       |
| status | int(11)     | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)

mysql> select * from leave_form;
+---------------+-------------+--------+--------+------------+------+------------+--------+
| name          | regno       | A_id   | H_id   | date       | days | reason     | status |
+---------------+-------------+--------+--------+------------+------+------------+--------+
| S.Lavanya     | 31113104702 | 104002 | 104020 | 0000-00-00 |    3 | fever      |      0 |
| Hannah Andrew | 31113104025 | 104003 | 104030 | 0000-00-00 |    5 | tooth-ache |      0 |
+---------------+-------------+--------+--------+------------+------+------------+--------+
2 rows in set (0.00 sec)

mysql> select * from circular;
+------------+-----------------------------------------+
| date       | content                                 |
+------------+-----------------------------------------+
| 0000-00-00 | Tomorrow licet remains closed           |
| 2017-03-16 | Tomorrow licet remains closed           |
| 2017-03-16 | students requested to pay exam fees by today |
+------------+-----------------------------------------+
3 rows in set (0.00 sec)

mysql> █
```

Figure 7.6 Leave form table and circular description

2.3 Screenshots – Android Application



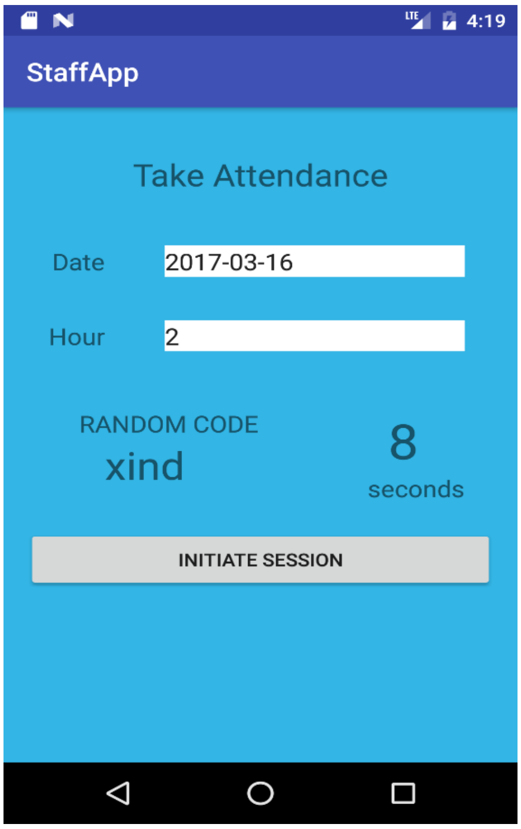
StaffApp

LICET Staff App

Username adv_ece

Password •••••••••

Figure 7.7 Staff login



StaffApp

Take Attendance

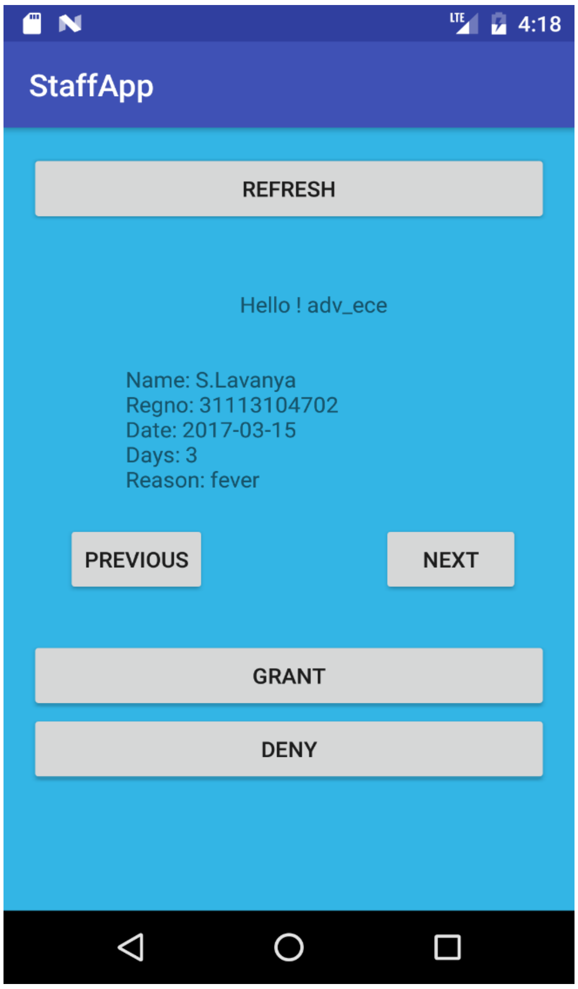
Date 2017-03-16

Hour 2

RANDOM CODE xind

8 seconds

Figure 7.8 Attendance session



Hello ! adv_ece

Name: S.Lavanya
Regno: 31113104702
Date: 2017-03-15
Days: 3
Reason: fever

Figure 7.9 Handling student requests

REFERENCES

1. Andi Adriansyah; and Akhmad Wahyu Dani (2014) ‘Design of Small Smart Home System’
2. L.A. Gama-Moreno; A. Corralejo; A. Ramirez-Molina; J. A. Torres-Rangel; C. Martinez-Hernandez; and M.A. Juarez (2016) ‘A Design of a Water Tanks Monitoring System Based on Mobile Devices’
3. Mohammad Shoeb Shah; and P. B. Borole (2016) ‘Surveillance and Rescue Robot using Android’
4. Sanja Maravić Čisar; Robert Pinter; Viktor Vojnić; Vanja Tumbas; and Petar Čisar (2016) ‘Smartphone Application for Tracking Students’ Class Attendance’
5. Shariq Suhail; Viswanatha Reddy; Rambabu; Dharma Savarni C. V.; and V. K. Mittal5 (2016) ‘Multi-Functional Secured Smart Home’
6. *https://www.w3schools.com/php/php_mysql_select.asp* - PHP to get data from the database.
7. *http://stackoverflow.com/questions/8687611/efficient-way-to-send-data-from-android-device-to-apache-web-server* - android to server mechanisms.
8. *http://bohdan-danishevsky.blogspot.in/2017/01/building-python-360-on-raspberry-pi-3.html* - python installation in raspberry pi.
9. *https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu* - LAMP server installation in Ubuntu 15.04.
10. *https://www.simplifiedcoding.net/android-studio-volley-tutorial-to-create-a-login-application/* - android code reference for mobile application.
11. *http://www.expertreviews.co.uk/raspberry-pi-foundation/1404450/raspberry-pi-3-review-wi-fi-tests-and-benchmarks* - Raspberry Pi 3 review - Wi-Fi tests and benchmarks.