ADVANCE WIRELESS NOTICE BOARD

Submitted in partial fulfillment of the requirements

for the degree of

Bachelor of Engineering

by

RUSHIKESH PURUSHOTTAM BELEKAR

05

MAKARAND SAMIR KHADAKBAN

30

RAHUL RAVINDRA SHINDE

64

Under the Supervision of

PROF. A. D. PALSODKAR

DEPARTMENT OF INFORMATION TECHNOLOGY

KONKAN GYANPEETH COLLEGE OF ENGINEERING

KARJAT-410201

April 2022

Certificate

This is to certify that the project entitled **ADVANCE WIRELESS NOTICE BOARD** is a bonafide work of **RUSHIKESH PURUSHOTTAM BELEKAR(05), MAKARAND SAMIR KHADAKBAN(30), RAHUL RAVINDRA SHINDE(64)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Undergraduate** in **DEPARTMENT OF INFORMATION TECHNOLOGY**.

Supervisor/Guide

Professor
A. D. PALSODKAR
Department of Information Technology

Head of Department

Professor
A. S. KUNTE
Department of Information Technology

Principal

Dr.
V. J. PILLEWAN
Konkan Gyanpeeth College of Engineering

Project Report Approval for B.E.

This thesis / dissertation/project report entitled **ADVANCE WIRELESS NOTICE BOARD** by **RUSHIKESH PURUSHOTTAM BELEKAR(05), MAKARAND SAMIR KHADAKBAN(30), RAHUL RAVINDRA SHINDE(64)** is approved for the degree of **DEPARTMENT OF INFORMATION TECHNOLOGY**.

Examiners

1.....

2.....

Date.

Place.

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature
(RUSHIKESH PURUSHOTTAM BELEKAR) 05

Signature
(MAKARAND SAMIR KHADAKBAN) 30

Signature
(RAHUL RAVINDRA SHINDE) 64

Date.

Abstract

In any organization or institution a Notice Board is a very important component as it is one of the best medium to communicate with the individuals of that organization or institution. The System ADVANCE WIRELESS NOTICE BOARD using a Raspberry Pi 4 microcontroller in a digital display system which has been designed to display a notice in a very attractive manner. The system allows the user to display the message (text) from anywhere just by sending the text via Android application. The sent text is received at the Raspberry Pi 4 microcontroller and enables it to get displayed on LCD screen. The system is an experiment for displaying real time notices.

Acknowledgements

This project work has been the most practical and exciting part of my learning experience, which would be an asset for future career. No system is created entirely by an individual. Many people have helped to create this system and each of their contribution has been valuable. Proper organization of concept and analysis of the system is due to keen interest and helping hand of my teacher and colleagues. Our sincere thanks to our project in charge Prof. A. D. Palsodkar for molding our thoughts and vision towards all subjects, we were studying in all three years. We are deeply thankful to Prof. A. S. Kunte, Head of our Department, who was a constant source of inspiration not only during this project, but also in last three years, spend in pursuing our degree at our college. We are also greatly thankful to the entire Information Technology Laboratory Staff who have helped us in completion of this project directly or indirectly. Last but not the least we would like to thank our parents and our college friends who helped us and whose good wishes and backings gave us strong support. We have put in our best efforts to make this project valuable and easy handling and operating will make it successful.

Contents

List of Figures

Abbreviations

SDK	**S**oftware **D**evelopment **K**it
IDE	**I**ntegrated **D**evelopment **E**nvironment
HDMI	**H**i-**D**efinition **M**ultimedia **I**nterface
API	**A**pplication **P**rogramming **I**nterface

Chapter 1

INTRODUCTION

1.1 Introduction

Nowadays conveying messages at large using notice boards, are widely used ones ranging from schools to organizations. We know the significance of notice boards in public areas like bus stands, railway stations, airports and banks, etc. But day-to-day changing of these boards is a very difficult task and a waste of time. At present, all electronic boards are designed with a wired system. The major drawback of designing these boards is that they are not flexible and cannot be located anywhere due to messy wire. To overcome this problem, a wireless board is designed to display the latest information. This article gives you an overview of how to design a wireless notice board using cloud technology. This notice board displays the information on Dot Matrix display (the text) you sent from the mobile app.

1.2 Objectives

Due to the popularity of internet, we choose internet as a medium for transferring information. The Internet of things (IoT) is the network of physical devices, vehicles, home appliances and other items embedded with electronics and software which enables these objects to connect and exchange data. Each device is uniquely identifiable through its

Embedded computing system but is able to inter operate within the existing Internet infrastructure. To provide security, we add username and password type authentication system. So only respective authority can send information. Raspberry pi which is the Heart of our system. A monitor is interfaced with Raspberry Pi. So information in the form of text can be displayed on the large screens.

1.3 Purpose, Scope, and Applicability

Purpose, Scope and Applicability: The description of Purpose, Scope, and Applicability are given below:

1.3.1 Purpose

Our primary aim is to get more people's attention on the display. By the usage of high definition display devices people can get more attention on the notice board rather than conventional notice boards. The conventional wireless notice board can display only texted messages from limited distance. But in our newly implemented system we can display text messages anywhere from the globe.

1.3.2 Scope

Notice boards are the places to leave public information such as advertise events, announce events or provide attention to the public, etc. Nowadays a separate person is needed to stick those information on the notice board. It will lead to loss of time as well as usage of manpower. The problems faced by the wooden or conventional type notice boards are resolved by the implementation of our ADVANCE WIRELESS NOTICE BOARD. It will bring an advanced means of passing notices around in the world in a much easier and efficient way.

1.3.3 Applicability

In Educational institutions majority of information given from the higher authorities in the form of text notice format. So displaying this types of information makes our system more user friendly. Due to the utilization of internet the sender can send message anywhere in the world. There is no range limitation for the successful exchange of information.

1.4 Achievements

Now the world is moving towards automation, so in this world if we want to do some changes in the previously used system we have to use the new techniques. Wireless operation provides fast transmission over long range communication. It saves resources and time. Data can be sent from remote location. User authentication is provided so that unknown person cannot access this notice board is a secured one. Previously the notice board using GSM were used and there was the limit of messages but in our system multiple text messages data can be stored on cloud storage. Text messages data can be seen whenever we want to see is an advantage of using our project.

Chapter 2

LITERATURE SURVEY

GSM Based Wireless Notice Board(March 2016)

Prof. Ravindra Joshi, Abhishek Gupta, Rani Borkar, Samita Gawas, Sarang Joshi.
This paper describes the design and construction of E-notice board using GSM technology. The system cosists of four basic units: GSM modem, Raspberry pi board, LCD monitor and Mobile device.
The operation of the system is centered on Raspberrypi Board. The operation of system is such that the notice which is to be displayed is send by the mobile device to the GSM modem and displayed on the LCD monitor using Raspberry Pi board. The system is based on real time process and saves lot of resources i.e. human effort. The main objective of this paper is to develop a wireless e-notice board that displays message sent from the user and to design a simple, easy to install, user friendly system, user friendly system. Wi-Fi provides higher data rates for multimedia access as compared to bluetooth which provides lower data transfer rates. Bluetooth are intended for communication (about 10m), while Wi-Fi is designed for WLAN about 100.
But when using GSM we cannot display message without Network connectivity.

Chapter 3

REQUIREMENTS AND ANALYSIS

3.1 Problem Definition

Nowadays conveying messages at large using notice boards, are widely used ones ranging from schools to organizations. We know the significance of notice boards in public areas like bus stands, railway stations, airports and banks, etc. But day-to-day changing of these boards is a very difficult task and a waste of time. At present, all electronic boards are designed with a wired system. The major drawback of designing these boards is that they are not flexible and cannot be located anywhere due to messy wire.

3.2 Requirements Specification

3.2.1 Raspberry Pi 4

Raspberry Pi 4 Model B is the latest product in the popular Raspberry Pi range of computers. It offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+, while retaining backwards compatibility and similar power consumption. For the end user,

Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems.

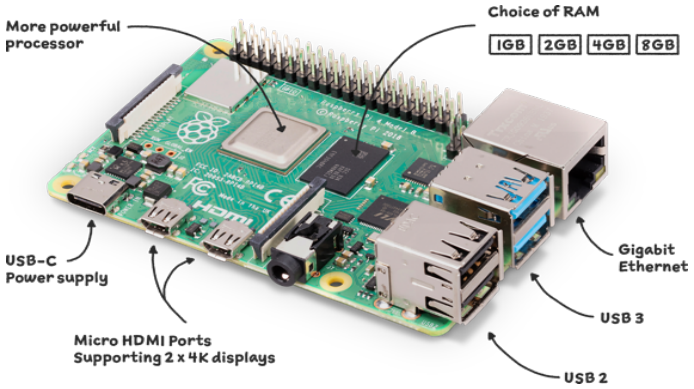

FIGURE 3.1: Raspberry Pi 4

This product's key features include a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI ports, hardware video decode at up to 4Kp60, up to 4GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on).

The dual-band wireless LAN and Bluetooth have modular compliance certification, allowing the board to be designed into end products with significantly reduced compliance testing, improving both cost and time to market.

3.2.2 Dot-Matrix Display 8*8

Dot matrix LED display contains the group of LEDs as a two dimensional array. They can display different types of characters or a group of characters. Dot matrix display is manufactured in various dimensions. Arrangement of LEDs in the matrix pattern is made in either of the two ways: Row anode-column cathode or Row cathode-column anode. By using this dot matrix display we can reduce the number of pins required for controlling all the LEDs.

FIGURE 3.2: Dot-Matrix Display

3.2.3 Jumper Wires(Female-To-Female)

FIGURE 3.3: Jumper Wires

A jumper wire is a short insulated wire with bare (stripped of insulation) ends. You use jumper wires, such as the one shown in the above figure, to connect two points in a breadboard circuit. Even if you have a set of precut jumper wires, chances are you'll need to make a jumper wire of a specific length for a circuit or two.

Premium Female to Female Jumper Wires - 40 x 8" (200mm).Female connectors on both side for microcontroller, arduino and sensor connections 40 different wires with 4 pieces of each of 10 different rainbow colors. Standard 0.1" (2.54mm) pitch. Each wire is 28 AWG.

3.2.4 Android Studio

FIGURE 3.4: Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.[8] It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android software development is the process by which new applications are created for devices running the Android operating system. Google states that"Android apps can be written using Kotlin, Java, and C++ languages" using the Android software development kit(SDK), while using other languages is also possible. All non-JVM languages, such as Go, JavaScript, C, C++ or assembly, need the help of JVM language code, that may be supplied by tools, likely with restricted API support. Some languages/programming tools allow cross-platform app support, i.e. for both Android and iOS. Third party tools, development environments and language support have also continued to evolve and expand since the initial SDK was released in 2008.

In addition, with major business entities like Walmart, Amazon, Bank of America etc. eyeing to engage and sell through mobiles, mobile application development is witnessing a transformation.

3.2.5 Raspbian OS

FIGURE 3.5: Raspbian OS

Raspberry Pi OS (formerly Raspbian) is a Debian-based operating system for Raspberry Pi. Since 2013, it has been officially provided by the Raspberry Pi Foundation as the primary operating system for the Raspberry Pi family of compact single-board computers. Raspberry Pi OS was first developed by Mike Thompson and Peter Green as Raspbian, an independent and unofficial port of Debian to the Raspberry Pi. The first build was released on July 15, 2012. As the Raspberry Pi had no officially provided operating system at the time, the Raspberry Pi Foundation decided to build off of the work done by the Raspbian project and began producing and releasing their own version of the software. The Foundation's first release of Raspbian, which now referred both to the community project as well as the official operating system, was announced on September 10th, 2013. On May 28th, 2020, the Raspberry Pi Foundation announced they were releasing a beta 64-bit version of their official operating system. However, the 64-bit version was not based on Raspbian, instead taking its userland from Debian directly. Since the Foundation did not want to use the name Raspbian to refer to software that was not based on the Raspbian project, the name of the officially provided operating system was changed to Raspberry Pi OS. This change was carried over to the 32-bit version as well, though it continued to be based on Raspbian. The 64-bit version of Raspberry Pi OS was officially released on February 2nd, 2022.

3.2.6 Buzzer

FIGURE 3.6: Buzzer

A buzzer or beeper is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric (piezo for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.

3.3 Software and Hardware Requirements

Following are the details of all the software and hardware needed for the development and implementation of our project.

3.3.1 Hardware Requirements

- Equipment: Raspberry Pi 4 microcontroller, Dot-Matrix Display(8*8)
- Graphics Card: GTX 710
- Processor: I3 5th gen
- Disk Capacity: 120gb SSD/500gb HDD
- RAM Capacity: 8gb RAM DDR4

3.3.2 Software Requirements

- Android Studio: Latest version

- Application: Wireless Notice Board
- Testing Tool: AVD(Android Virtual Device)
- Development Environment Setup: Android SDK 5+, Rasbian OS

3.4 Preliminary Product Description

Existing system's objective is to send notice (text), from anywhere but it requires SIM which is based on GSM module. And it also uses messaging (SMS), which is outdated. Instead of using GSM, our system requires internet to send and receive the data and also to store the data.

3.5 Planning and Scheduling

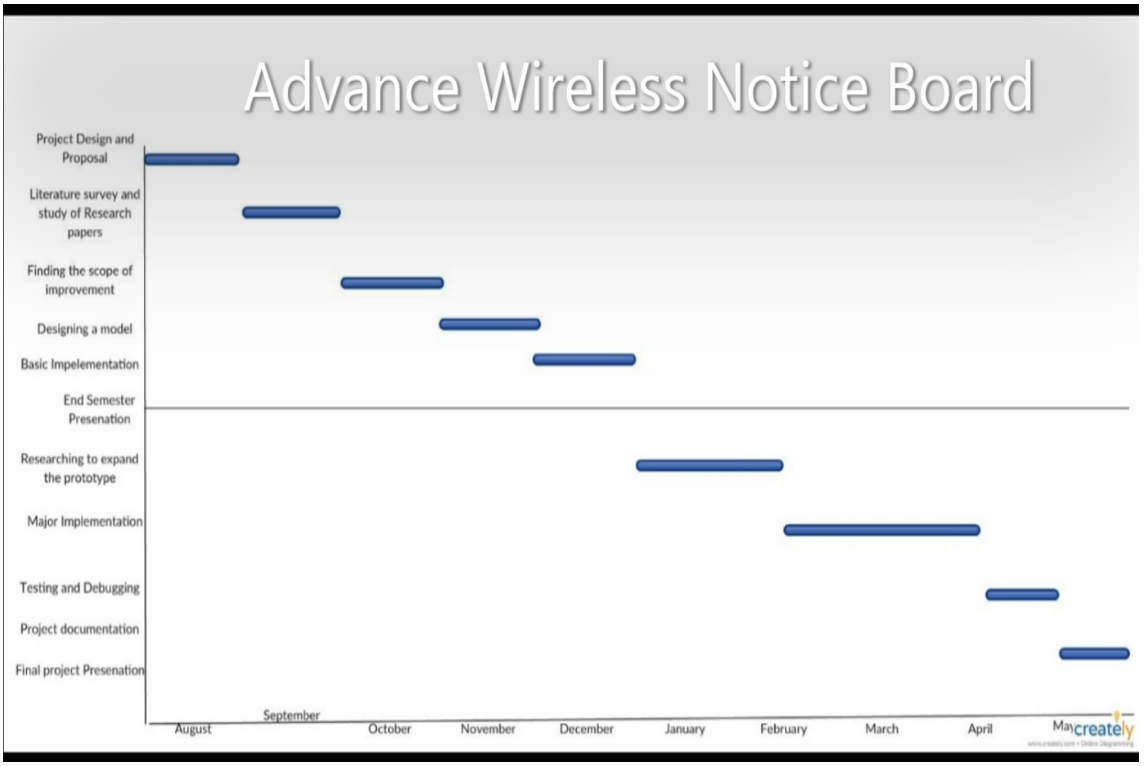

FIGURE 3.7: Gantt Chart

Chapter 4

SYSTEM DESIGN

Describes desired features and operations in detail, including screen layouts, business rules, process diagrams, pseudo code and other documentation.

4.1 Basic Modules

4.1.1 Android Software Development

Android software development is the process by which applications are created for devices running the Android operating system. Google states that "Android apps can be written using Kotlin, Java, and C++ languages" using the Android software development kit (SDK), while using other languages is also possible. All non-Java virtual machine (JVM) languages, such as Go, JavaScript, C, C++ or assembly, need the help of JVM language code, that may be supplied by tools, likely with restricted API support. Some programming languages and tools allow cross-platform app support (i.e. for both Android and iOS). Third party tools, development environments, and language support have also continued to evolve and expand since the initial SDK was released in 2008. The official Android app distribution mechanism to end users is Google Play; it also allows staged gradual app release, as well as distribution of pre-release app versions to testers.

FIGURE 4.1: The Android Stack

The Android software development kit (SDK) includes a comprehensive set of development tools. The Android SDK Platform Tools are a separately downloadable subset of the full SDK, consisting of command-line tools such as adb and fastboot. The Android Debug Bridge (ADB) is a tool to run commands on a connected Android device. Fastboot is a protocol used for flashing filesystems. Code written in C/C++ can be compiled to ARM, or x86 native code (or their 64-bit variants) using the Android Native Development Kit (NDK).

4.1.2 Cloud Computing

Cloud computing is when computing services are provided by a company or place outside of where they are being used. It is like the way in which electricity is sent to users: they simply use the electricity that is sent to them and do not need to worry where the electricity is from or how it is made and brought to them. Every month, they pay only for what they used and nothing more. The idea behind cloud computing is similar: The user can simply use storage, computing power, or development environments, without having to worry how they work behind the scenes.

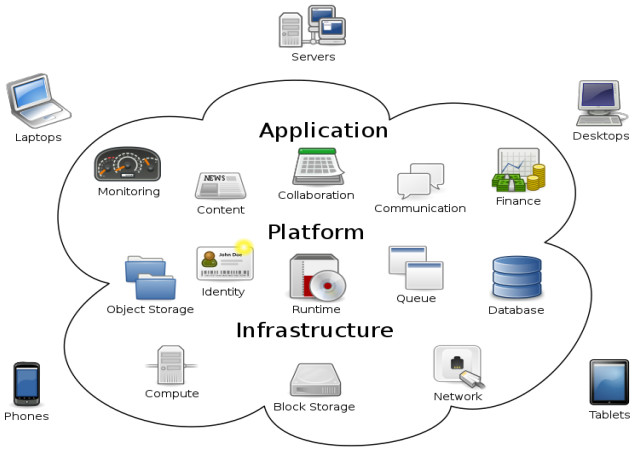

FIGURE 4.2: Cloud Computing

The cloud is a metaphor for the Internet based on how it is described in computer network diagrams. Just as how in the real world, clouds hide parts of the sky from sight, the cloud in computing hides the complex infrastructure that makes the Internet work. It is a type of computing in which IT-related actions are provided "as a service", allowing users to access these services through the Internet ("in the cloud"). They do not have to know or control the technologies behind them, preventing them from running into ethical and legal problems.

According to the IEEE, cloud computing is a concept where information is placed on servers and sent over the Internet to other devices, such as computers, laptops, handhelds, and sensors. It includes the idea of having software as a service (SaaS), such as Web 2.0, that depend on the Internet to meet the needs of their users. For example, Google has made several office suite apps which are accessed from a web browser. Unlike other software that does the same tasks, including Microsoft Office, the software and data are stored on Google's servers, not on the machine in which they are used.

4.1.3 Internt of Things(IoT)

The Internet of things (IoT) describes physical objects (or groups of such objects) with sensors, processing ability, software, and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks.

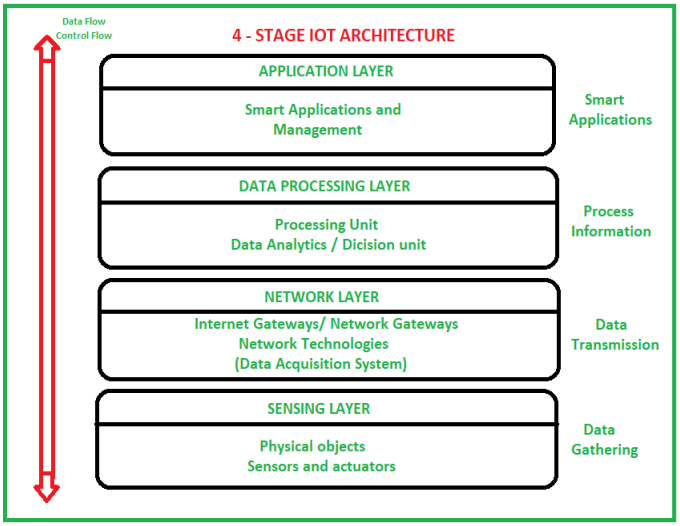

FIGURE 4.3: Architecture of IoT

Internet of things has been considered a misnomer because devices do not need to be connected to the public internet, they only need to be connected to a network and be individually addressable.

The field has evolved due to the convergence of multiple technologies, including ubiquitous computing, commodity sensors, increasingly powerful embedded systems, and machine learning. Traditional fields of embedded systems, wireless sensor networks, control systems, automation (including home and building automation), independently and collectively enable the Internet of things. In the consumer market, IoT technology is most synonymous with products pertaining to the concept of the "smart home", including devices and appliances (such as lighting fixtures, thermostats, home security systems, cameras, and other home appliances) that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smartphones and smart speakers. IoT is also used in healthcare systems.

There are a number of concerns about the risks in the growth of IoT technologies and products, especially in the areas of privacy and security, and consequently, industry and governmental moves to address these concerns have begun, including the development of international and local standards, guidelines, and regulatory frameworks.

4.2 Data Design

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data to the database model. Database management system manages the data accordingly.

Database design involves classifying data and identifying interrelationships. This theoretical representation of the data is called an ontology. The ontology is the theory behind the database's design.

In a majority of cases, a person who is doing the design of a database is a person with expertise in the area of database design, rather than expertise in the domain from which the data to be stored is drawn e.g. financial information, biological information etc. Therefore, the data to be stored in the database must be determined in cooperation with a person who does have expertise in that domain, and who is aware of what data must be stored within the system.

This process is one which is generally considered part of requirements analysis, and requires skill on the part of the database designer to elicit the needed information from those with the domain knowledge. This is because those with the necessary domain knowledge frequently cannot express clearly what their system requirements for the database are as they are unaccustomed to thinking in terms of the discrete data elements which must be stored. Data to be stored can be determined by Requirement Specification.

4.3 Procedural Design

Procedural modeling is an umbrella term for a number of techniques in computer graphics to create 3D models and textures from sets of rules. L-Systems, fractals, and generative modeling are procedural modeling techniques since they apply algorithms for producing scenes. The set of rules may either be embedded into the algorithm, configurable by parameters, or the set of rules is separate from the evaluation engine. The output is called procedural content, which can be used in computer games, films, be uploaded to the internet, or the user may edit the content manually.

Procedural models often exhibit database amplification, meaning that large scenes can be generated from a much smaller number of rules. If the employed algorithm produces the same output every time, the output need not be stored. Often, it suffices to start the algorithm with the same random seed to achieve this.

Although all modeling techniques on a computer require algorithms to manage and store data at some point, procedural modeling focuses on creating a model from a rule set, rather than editing the model via user input. Procedural modeling is often applied when it would be too cumbersome to create a 3D model using generic 3D modelers, or when more specialized tools are required. This is often the case for plants, architecture or landscapes.

4.3.1 Block Diagram

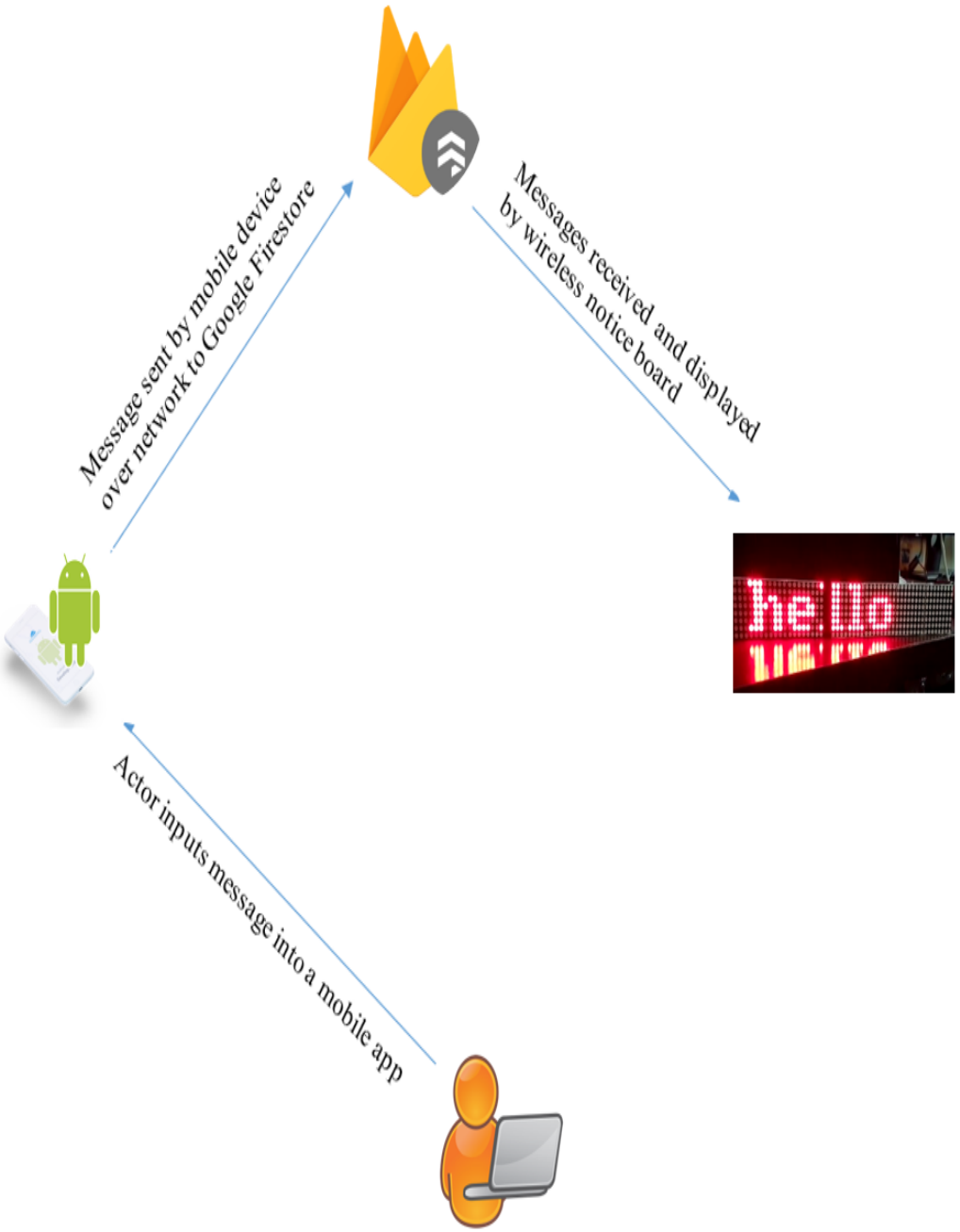

FIGURE 4.4: Block Diagram

4.3.2 Flow Chart Diagram

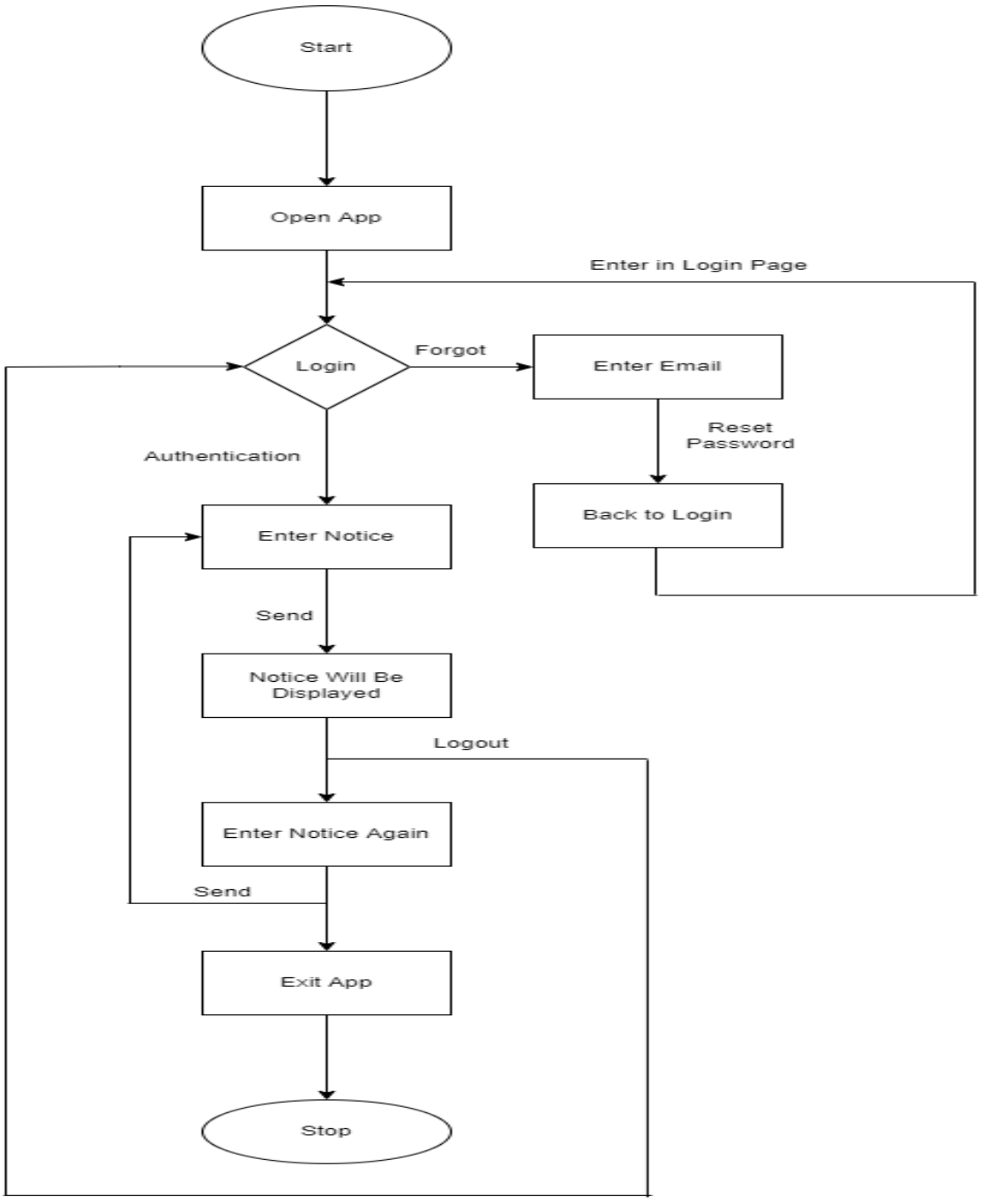

FIGURE 4.5: Flow Chart Diagram

4.3.3 UseCase Diagram

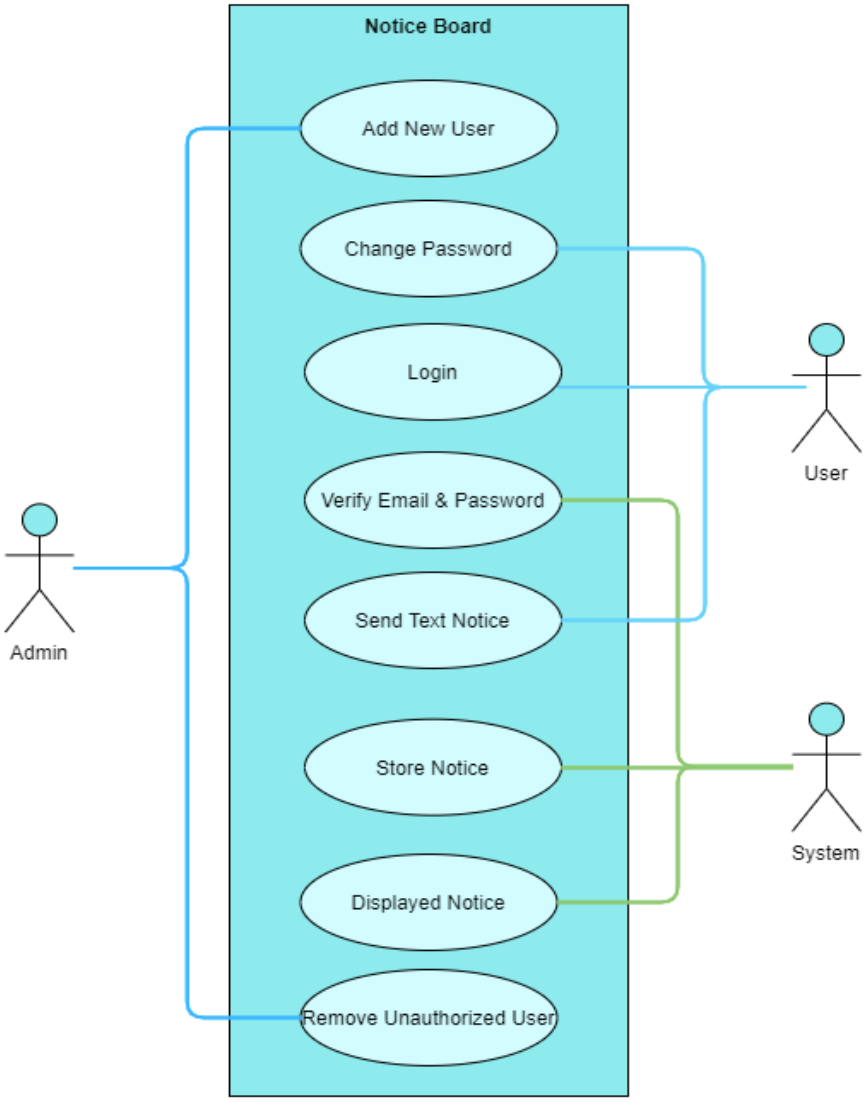

FIGURE 4.6: UseCase Diagram

4.3.4 Class Diagram

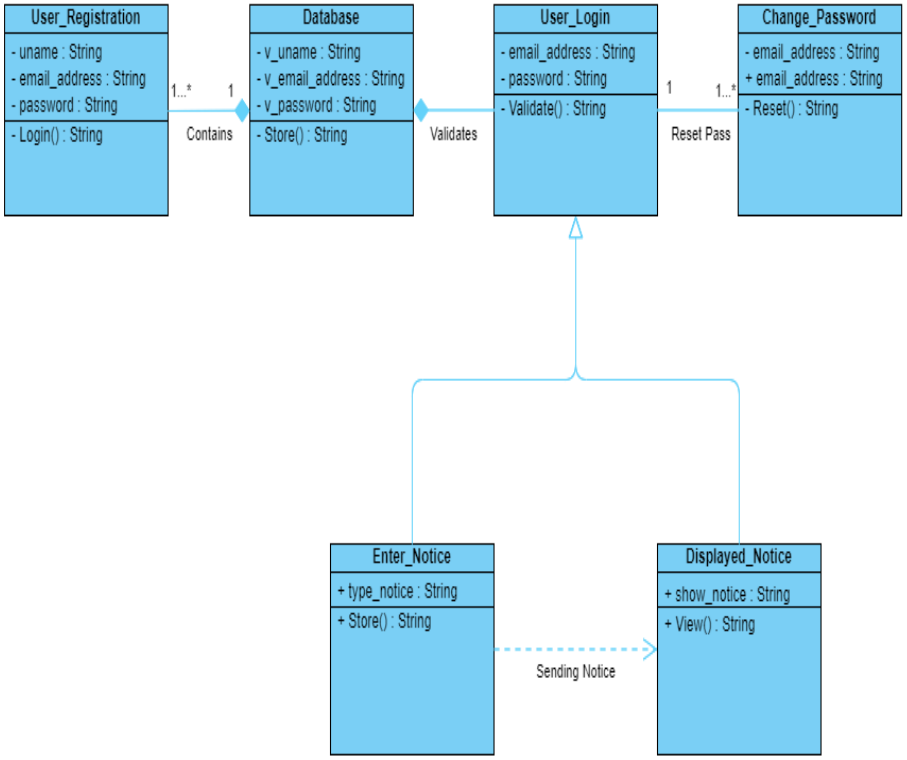

FIGURE 4.7: Class Diagram

4.3.5 Sequence Diagram

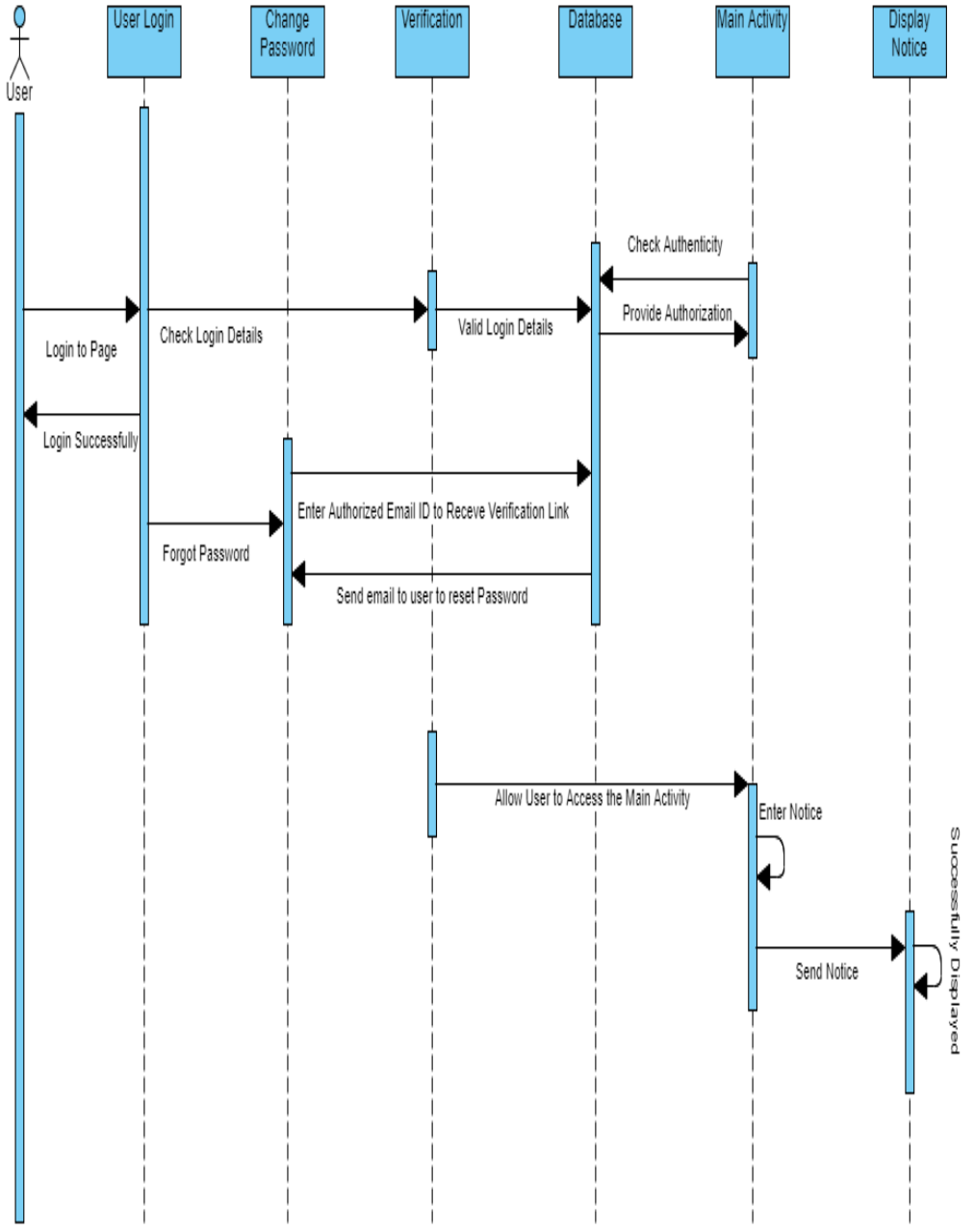

FIGURE 4.8: Sequence Diagram

4.3.6 Pinout Diagram

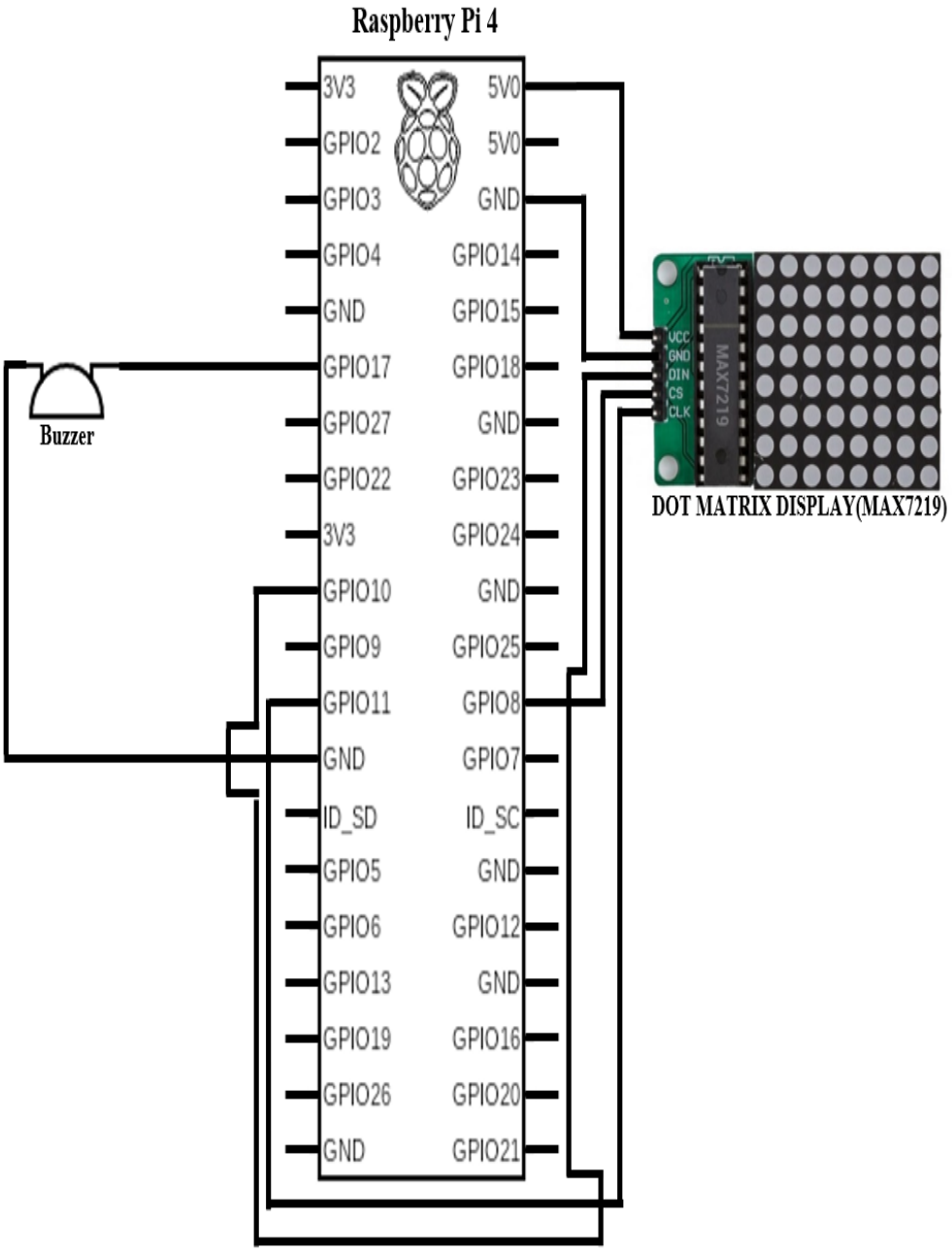

FIGURE 4.9: Pinout Diagram

4.4 Security Issues

As our project is running on Google services we actually don't have any security issues but to add extra layer of security to our project we only provide access to authorised users. This way our project is more secure and efficient.

4.5 Test Cases Design

- **System Environment**

  Input power: 5V DC via USB-C connector, 5V DC via GPIO header, Power over Ethernet (PoE)–enabled

  Environment: Operating temperature 0–50°C

- **Mobile Application Environment**

  Internet Connection: You must have a stable and fast internet connection to ensure proper working of system.

  Android Version: You must have device(smartphone) with minimum android version 5/5+

  Environment: Operating temperature 0–50°C

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Implementation Approaches

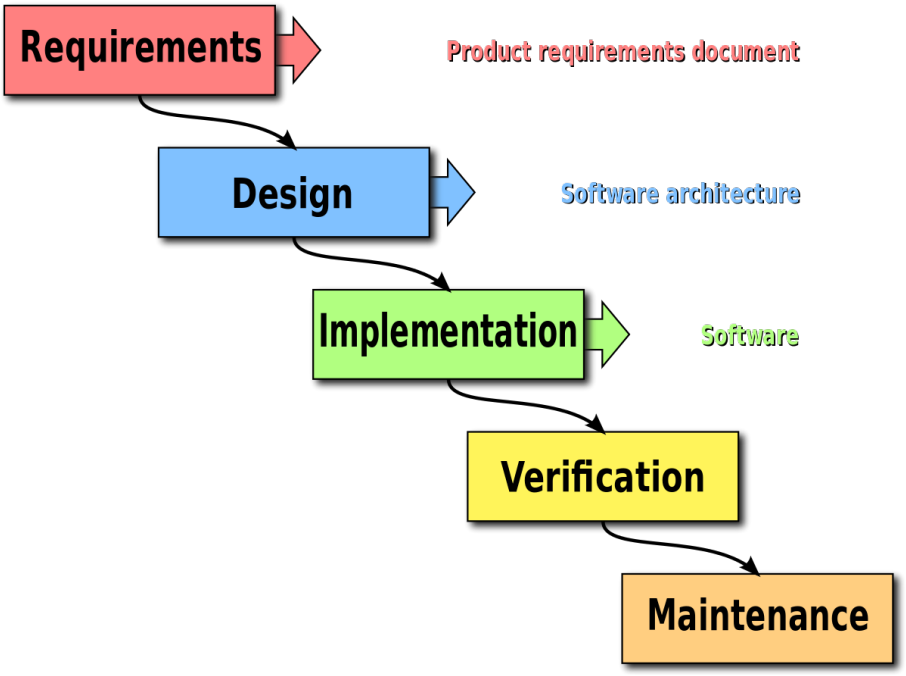

FIGURE 5.1: Waterfall Model

The waterfall model is a classical model used in system development life cycle to create a system with a linear and sequential approach. It is termed as waterfall because the model develops systematically from one phase to another in a downward fashion. This model is divided into different phases and the output of one phase is used as the input of the next phase. Every phase has to be completed before the next phase starts and there is no overlapping of the phases.
The waterfall methodology is composed of seven non-overlapping stages:

- Requirements: Potential requirements, deadlines and guidelines for the project are analyzed and placed into a functional specification. This stage handles the defining and planning of the project without mentioning specific processes.
- Analysis: The system specifications are analyzed to generate product models and business logic that will guide production. This is also when financial and technical resources are audited for feasibility.
- Design: A design specification document is created to outline technical design requirements such as programming language, hardware, data sources, architecture and services.
- Coding/Implementation: The source code is developed using the models, logic and requirements designated in the prior stages. Typically, the system is designed in smaller components, or units, before being implemented together.
- Testing: This is when quality assurance, unit, system and beta tests take place to report issues that may need to be resolved. This may cause a forced repeat of the coding stage for debugging. If the system passes the tests, the waterfall continues forward.
- Operation/Deployment: The product or application is deemed fully functional and is deployed to a live environment.
- Maintenance: Corrective, adaptive and perfective maintenance is carried out indefinitely to improve, update and enhance the final product. This could include releasing patch updates or releasing new versions.

5.1.1 Unit Testing

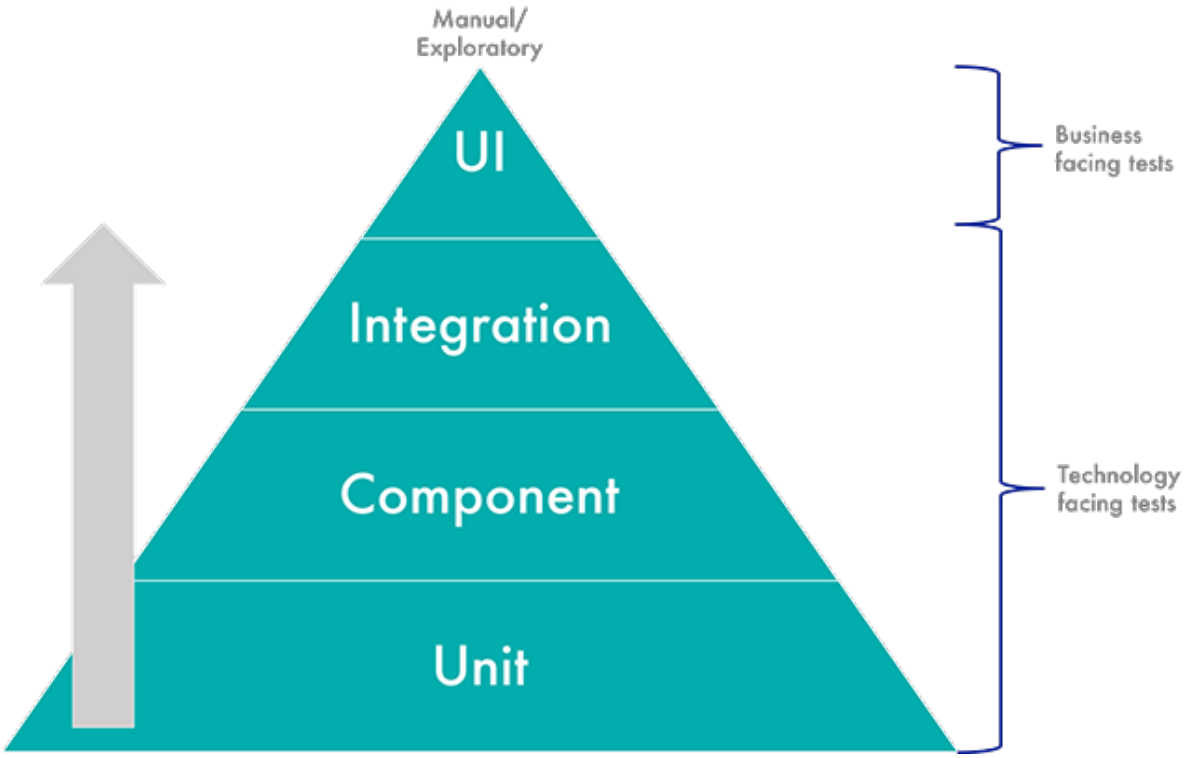

FIGURE 5.2: Unit Testing Model

In computer programming, unit testing is a software testing method by which individual units of source code—sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures—are tested to determine whether they are fit for use.

Unit tests are typically automated tests written and run by software developers to ensure that a section of an application (known as the "unit") meets its design and behaves as intended. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, or an individual method. By writing tests first for the smallest testable units, then the compound behaviors between those, one can build up comprehensive tests for complex applications.

To isolate issues that may arise, each test case should be tested independently. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation.

During development, a software developer may code criteria, or results that are known to be good, into the test to verify the unit's correctness. During test case execution, frameworks log tests that fail any criterion and report them in a summary. For this, the most commonly used approach is test - function - expected value.

5.2 Modifications and Improvements

Firstly we had developed project for offline use only. So we decided to make it complex by making it for online use so that user can access it anywhere from the globe.
In initial phase of development to make system online we used ESP8266 WIFI Module but capacity to send and receive data from cloud was very less and also ESP8266 runs on arduino system which requires Assembly Language Programming which limits the functionality of the code.
So we decided to make it more reliable, efficient, secure and robust. we changed the hardware from ESP8266 to Raspberry Pi 4. Raspberry Pi 4 not only has Python support but also have more security than ESP8266.
For database initially we used Google Firebase Realtime Database to send, receive and store the data. But eventually we came to know that it has messy database data structure which leads to confusion while reading the data. So we shifted to Google Firebase's new database which is Google Firestore which has clean data structure and more robust functionalities as compare to Realtime Database.

Chapter 6

RESULTS AND DISCUSSION

6.1 Test Reports

6.1.1 No Internet Connection

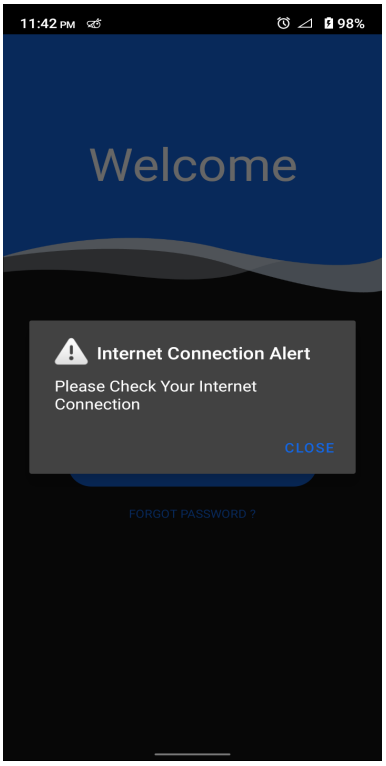

FIGURE 6.1: No Connection Snapshot

When you by mistake open an application without turning on mobile data you will see this type of error.

6.1.2 Empty Fields

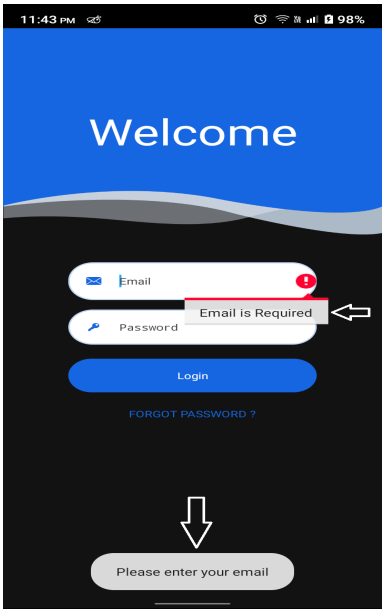

FIGURE 6.2: Empty Email Snapshot

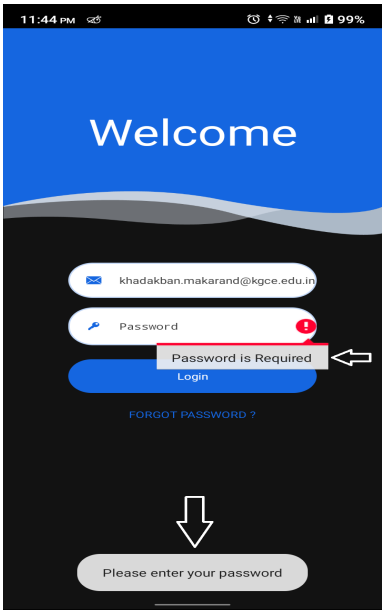

FIGURE 6.3: Empty Password Snapshot

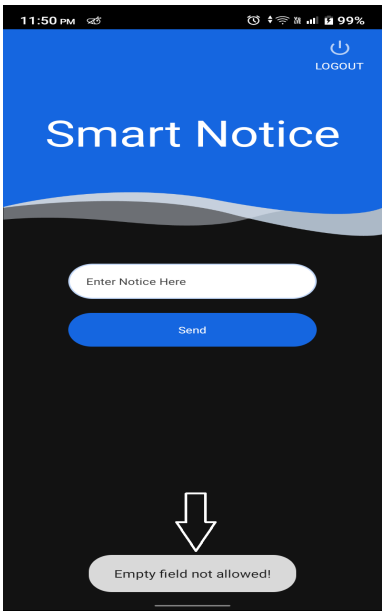

FIGURE 6.4: Empty Notice Snapshot

If you are trying to log in without entering all fields or if you try to send message/notice without entering in fields you will get these types of error.

6.1.3 White Spaces

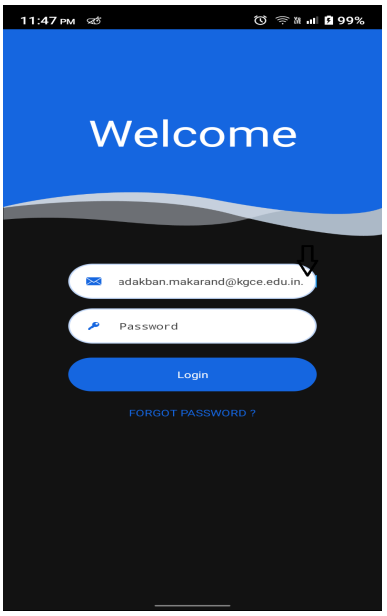

FIGURE 6.5: White Spaces Snapshot

If you accidentally add white spaces while entering credentials then do not worry, our team already took care of this problem.

6.1.4 Invalid/Incorrect Credentials

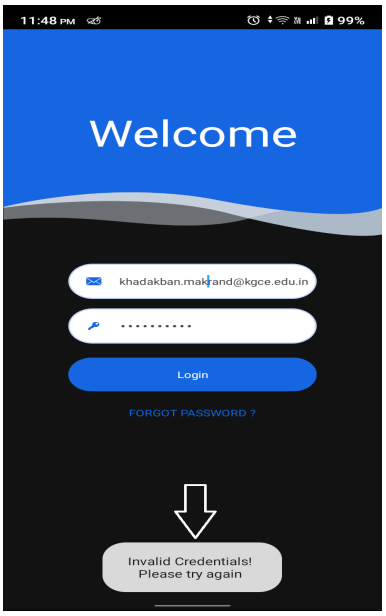

FIGURE 6.6: Invalid Credentials Snapshot

When you enter incorrect credentials in those fields you will get these types of error.

6.1.5 Alphanumeric Exception

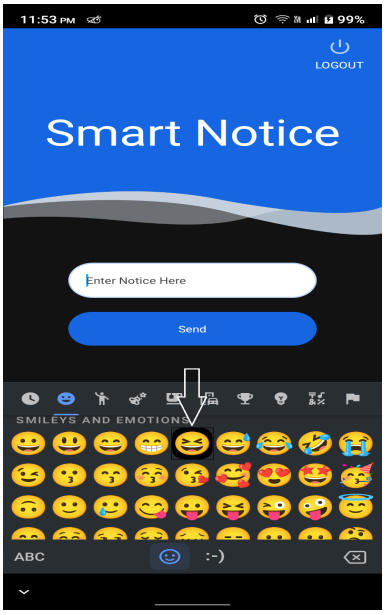

FIGURE 6.7: Emoji Exception Snapshot

When you try to enter emojis, languages other than English and $\leq$,$\geq$ *like this character you cannot type.*

FIGURE 6.8: Language Exception Snapshot

6.1.6 Acknowledgements

FIGURE 6.9: Before Sending Snapshot

As shown above and below, our application will ensure to keep you informed with the mistakes done by you in almost every scenario. So that you will not get panic or get confused while using our application and you will always be satisfied with user interface.

FIGURE 6.10: After Sending Snapshot

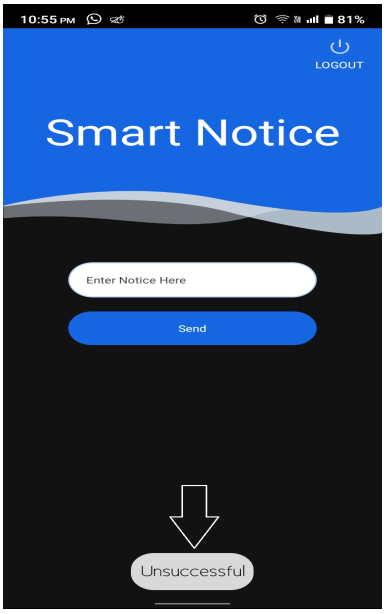

FIGURE 6.11: Unsuccessful Snapshot

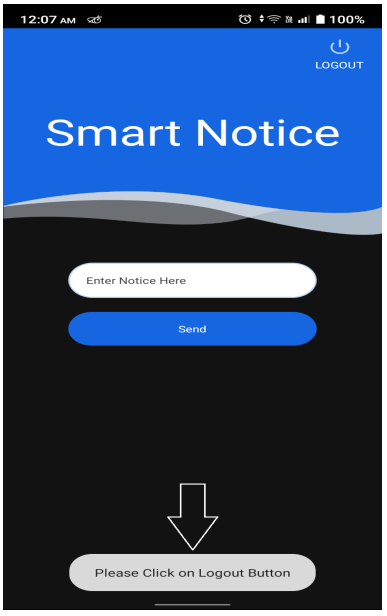

FIGURE 6.12: Back Button Home Page Snapshot

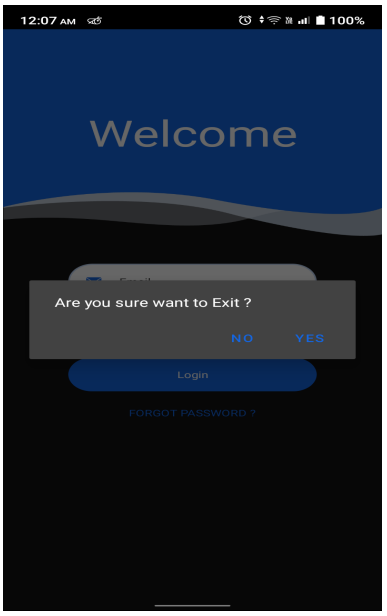

FIGURE 6.13: Back Button Login Page Snapshot

6.2 User Documentation

6.2.1 Step 1

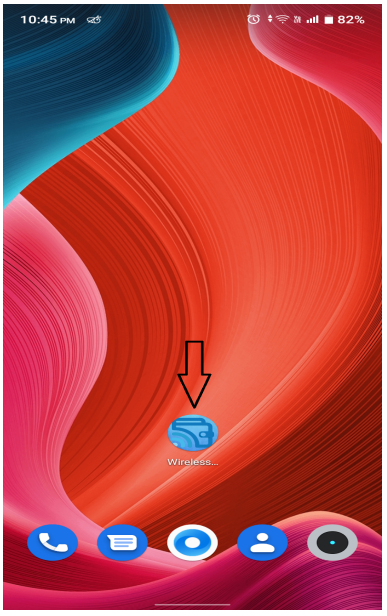

FIGURE 6.14: App Icon Snapshot

First you must ensure good internet connection availability to your device(smartphone). Then click on the icon of an app named as "Wireless Notice Board" to open the app

successfully.

6.2.2 Step 2

FIGURE 6.15: Splashscren Snapshot

As soon as you click on app icon you will notice nice app logo as a splash screen of an app which will load for about 2-3 seconds approximately.

6.2.3 Step 3

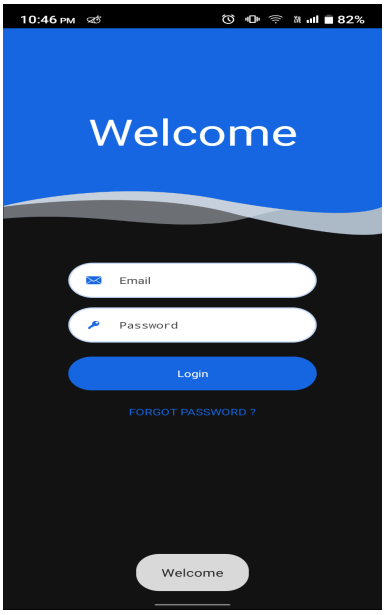

FIGURE 6.16: Login Page 1.1 Snapshot

As shown above, in next step you will see login page. This page is to provide user authentication.

6.2.4 Step 4

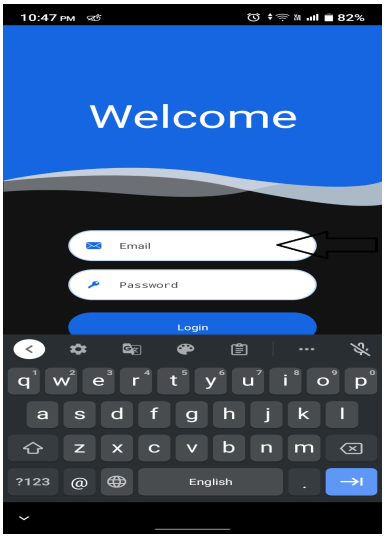

FIGURE 6.17: Login Page 1.2 Snapshot

Now please enter an email provided you as a credential by an admin; in the area shown in above figure.

6.2.5 Step 5

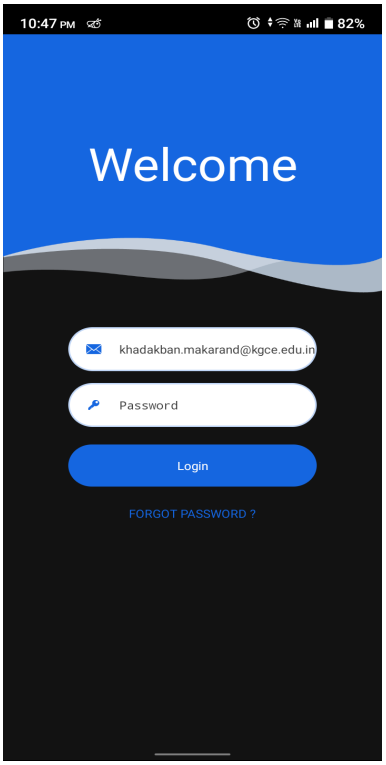

FIGURE 6.18: Login Page 1.3 Snapshot

Here is how it will look after you enter an email in the section provided specially for an email.

6.2.6 Step 6

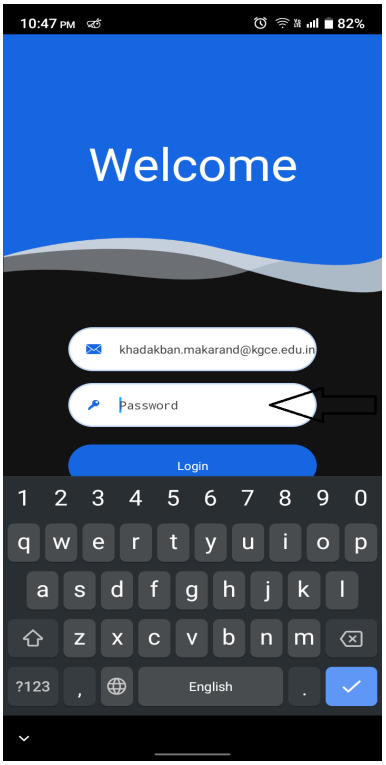

FIGURE 6.19: Login Page 1.4 Snapshot

Now please enter the password also provided you as a credential by an admin; in the area shown in above figure.

6.2.7 Step 7

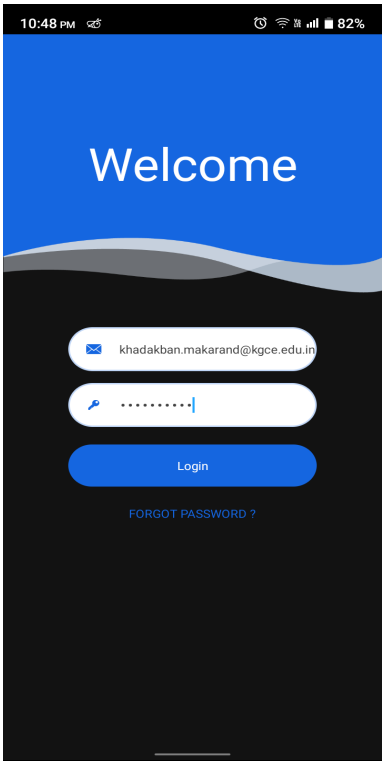

FIGURE 6.20: Login Page 1.5 Snapshot

Here is how it will look after you enter the password in the section provided specially for the password.

6.2.8 Step 8

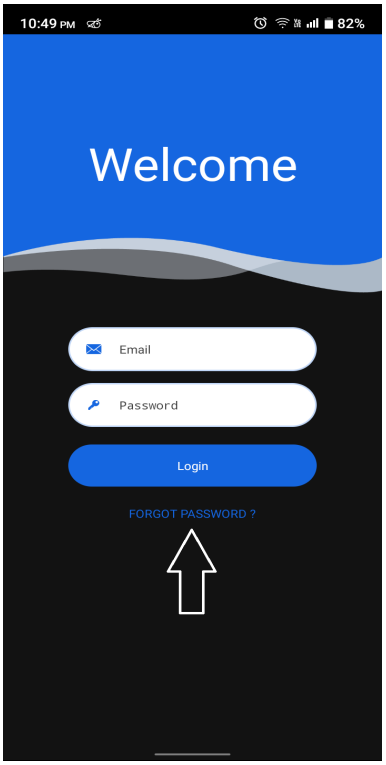

FIGURE 6.21: Fordot password Page 1.1 Snapshot

Go to step 13 if you know your credentials provided by an admin.

If you have forgotten the password provided by an admin then click on "Forgot Password?" button.

6.2.9 Step 9

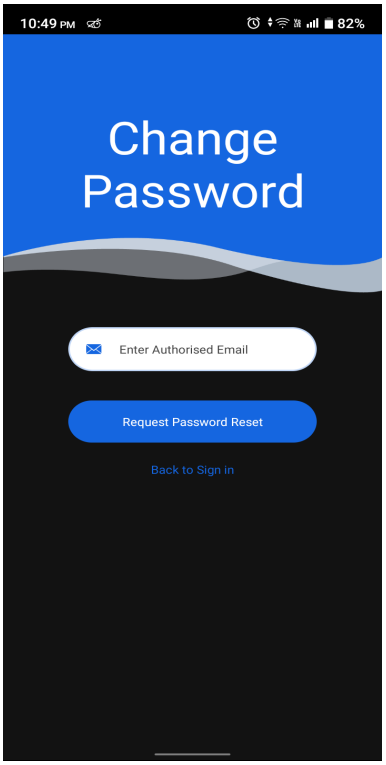

FIGURE 6.22: Fordot password Page 1.2 Snapshot

If you have clicked on "Forgot Password?" button by mistake then click on "Back to login" button to access Login Page again.

Now as you can see, you will land on "Change Password" section.

6.2.10 Step 10

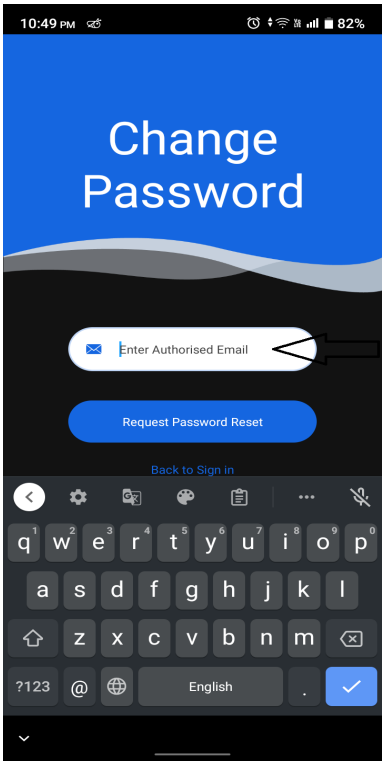

FIGURE 6.23: Fordot password Page 1.3 Snapshot

Enter your email provided by an admin as shown in figure above.

6.2.11 Step 11

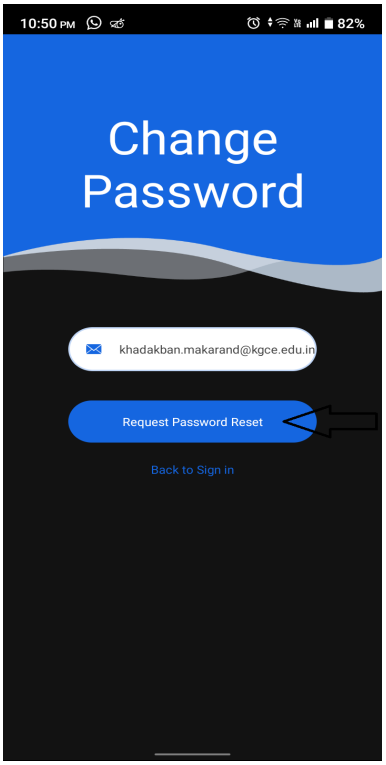

FIGURE 6.24: Fordot password Page 1.4 Snapshot

After entering your email, please click on "Request Password Reset" button.

6.2.12 Step 12

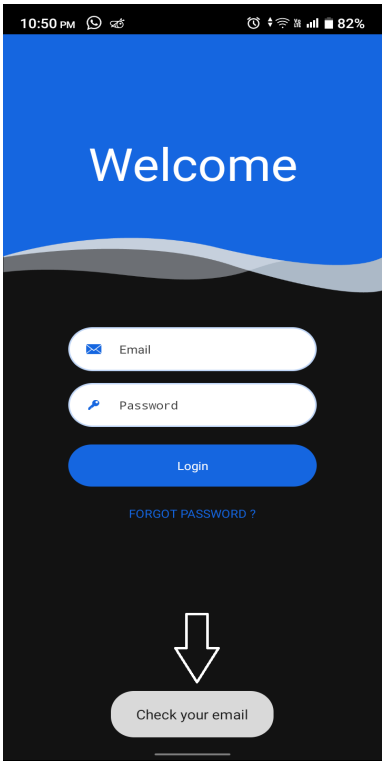

FIGURE 6.25: Login Page 1.6 Snapshot

Now as you received message "Check your email" please open your inbox to see if you have received any email from "Wireless Notice Board".

Then click on the link provided to change your password. After you successfully changed your password, please open our application again and your updated credentials as in above steps.

6.2.13 Step 13

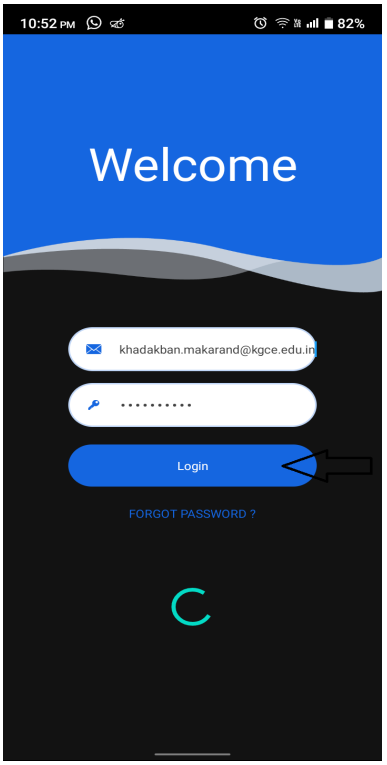

FIGURE 6.26: Login Page 1.7 Snapshot

After entering your credentials, please click on "Login" button to login into your account.

6.2.14 Step 14

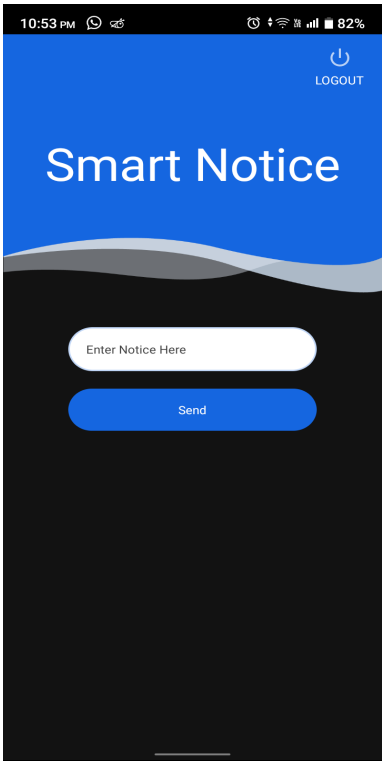

FIGURE 6.27: Home Page 1.1 Snapshot

Now as you can see, you will land on home page of our app.

6.2.15 Step 15

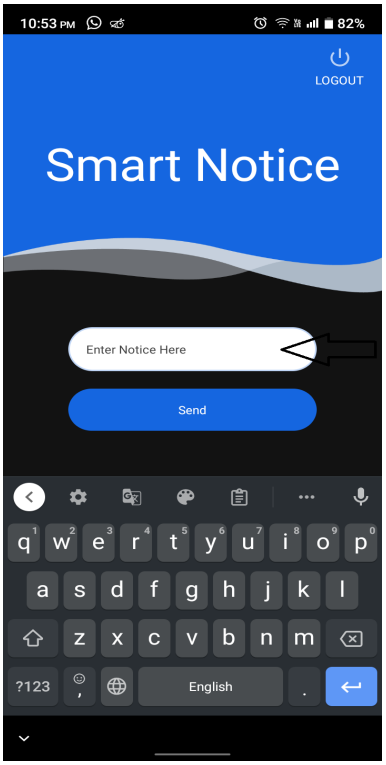

FIGURE 6.28: Home Page 1.2 Snapshot

Enter the message/notice you want to display on notice board in the section "Enter Notice Here" given below.

6.2.16 Step 16

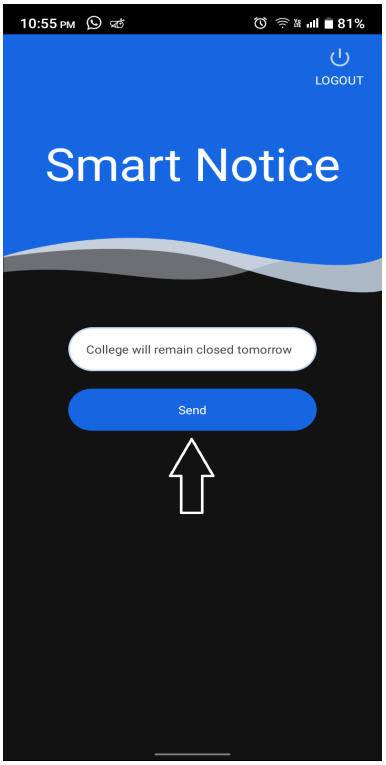

FIGURE 6.29: Home Page 1.3 Snapshot

Now click on "Send" button to display your notice.

6.2.17 Step 17

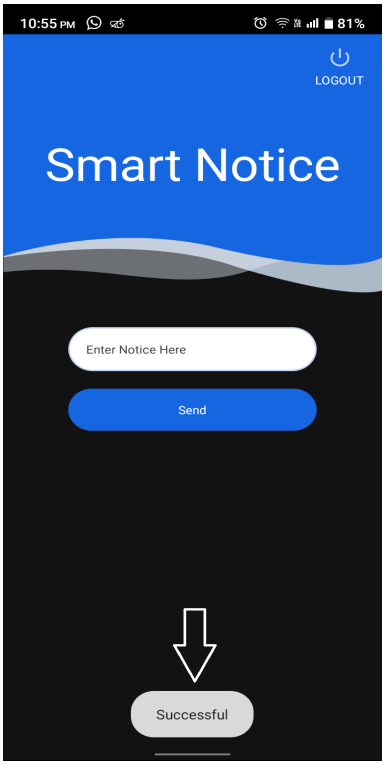

FIGURE 6.30: Home Page 1.4 Snapshot

You will see message as "Successful" that means the message you sent from an application is received successfully on notice board system and it will beep too so you can also get the confirmation and others can notice it too very easily. If previous message/notice is being displayed then your notice will be displayed shortly right after previous message/notice end. Now if you want to send another notice immediately you can do that as well and if you do not want to send another message/notice you can logout from an application by clicking on "LOGOUT" button.

6.2.18 Step 18

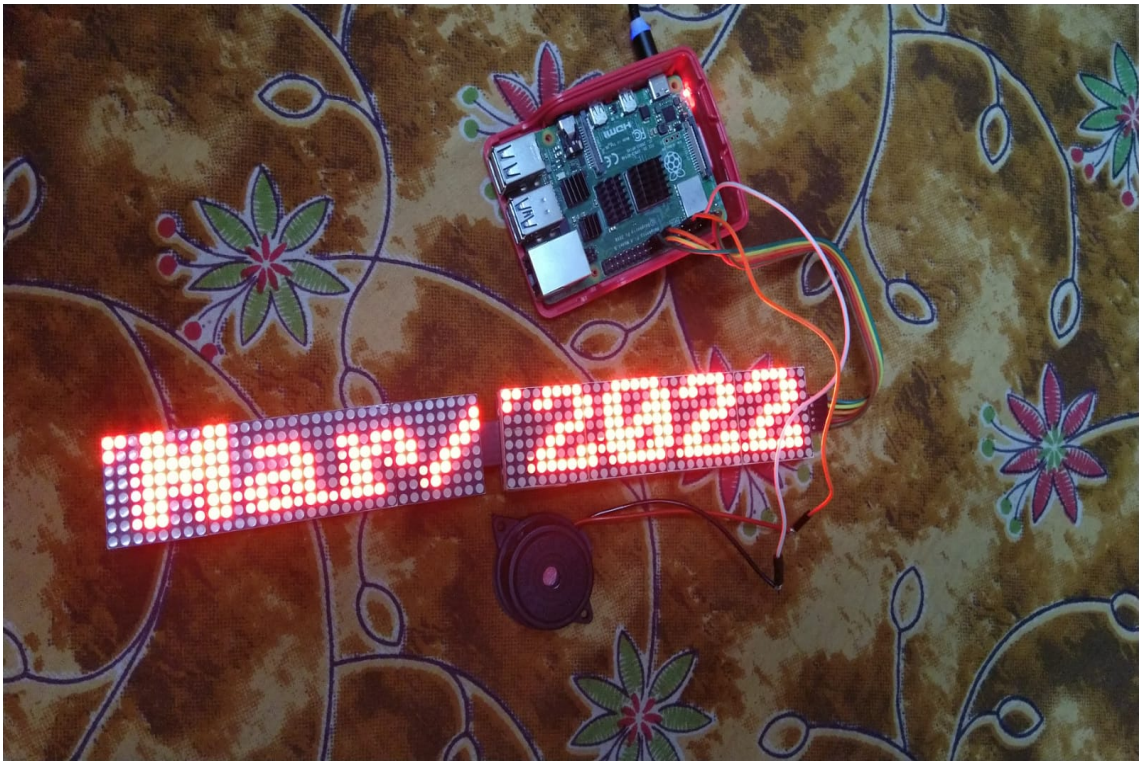

FIGURE 6.31: Output of Wireless Notice Board

Here we can see how it will look like.

Appendix A

AppendixA

CODE

A.1 Android Application Code

A.1.1 MainActivity.java

```
package com.example.v2;

import static android.widget.Toast.LENGTH_SHORT;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.text.TextUtils;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;
```

```
import java.lang.*;
import java.util.HashMap;
import java.util.Map;

// Connecting activities
public class MainActivity extends AppCompatActivity {

    FirebaseFirestore db = FirebaseFirestore.getInstance();
    @RequiresApi(api = Build.VERSION_CODES.O)

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

        // Connecting objects of fields with xml using findViewByID function
        // Creating objects of fields
        EditText enter = findViewById(R.id.Enter);
        TextView send = findViewById(R.id.sendBtn);

        // onClickListener for send button
        send.setOnClickListener(v -> {
            if (TextUtils.isEmpty(enter.getText().toString()))
            {
                Toast.makeText(MainActivity.this,
                        "Empty field not allowed!",
                        Toast.LENGTH_SHORT).show();
            } else {

                String text = enter.getText().toString();
                Intent mintent;
                mintent = getIntent();
                String uname = mintent.getStringExtra(Login.VSEND);

                // Using Hashmap to store data in Firestore
                long dt = System.currentTimeMillis();
                Map<String, Object> user = new HashMap<>();

                // Using put function to store data in Firestore
                user.put("Text", text);
                user.put("uid", uname);
                user.put("DT", dt);
                enter.getText().clear();

                db.collection("data").document()
```

```
                        .set(user)
                        .addOnSuccessListener(documentReference -> Toast
                        .makeText(getApplicationContext(),
                        "Successful", LENGTH_SHORT).show())
                        .addOnFailureListener(e -> Toast
                        .makeText(getApplicationContext(), "Unsuccessful",
                        LENGTH_SHORT).show());
            }
        });

        // Logout Button
        ImageButton logout = findViewById(R.id.signOut);
        logout.setOnClickListener(v -> {
            FirebaseAuth.getInstance().signOut();
            Intent mainActivity = new Intent(MainActivity.this, Login.class);
            mainActivity.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(mainActivity);
            finish();
        });

    }
    // Back Button
    @Override
    public void onBackPressed() {
        Toast.makeText(MainActivity.this, "Please Click on Logout Button",
        LENGTH_SHORT).show();
    }
}
```

A.1.2 Login.java

```
package com.example.v2;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Intent;
import android.content.Context;
import android.content.DialogInterface;
import android.net.ConnectivityManager;
import android.net.NetworkCapabilities;
import android.os.Build;
import android.os.Bundle;
import android.text.TextUtils;
```

```
import android.view.View;
import android.widget.ProgressBar;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;

import java.util.Objects;

public class Login extends Activity {

    public static final String VSEND = "com.example.v2.extra.NAME";

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

        EditText mEmail = findViewById(R.id.lEmail);
        EditText mPassword = findViewById(R.id.lPassword);
        TextView mLoginBtn = findViewById(R.id.loginBtn);
        TextView mPassBtn = findViewById(R.id.fPassword);
        ProgressBar progressBar = findViewById(R.id.progressBar);
        FirebaseAuth fAuth = FirebaseAuth.getInstance();

        // Login
        mLoginBtn.setOnClickListener(v -> {

            String email = mEmail.getText().toString().trim();
            String password = mPassword.getText().toString().trim();

            if(TextUtils.isEmpty(email)){
                Toast.makeText(Login.this, "Please enter your email",
                Toast.LENGTH_SHORT).show();
                mEmail.setError("Email is Required");
                return;
            }

            if(TextUtils.isEmpty(password)){
                Toast.makeText(Login.this, "Please enter your password",
                Toast.LENGTH_SHORT).show();
                mPassword.setError("Password is Required");
                return;
            }
```

```
        progressBar.setVisibility(View.VISIBLE);

        // authenticate the user
        fAuth.signInWithEmailAndPassword(email,password)
        .addOnCompleteListener(task ->
        {
            if(task.isSuccessful()){
                Toast.makeText(Login.this, "Logged in Successfully",
                Toast.LENGTH_SHORT).show();
                startActivity(new Intent(getApplicationContext()
                ,MainActivity.class));
                String uname = Objects.requireNonNull(fAuth.getCurrentUser())
                .getUid();
                Intent mintent = new Intent(Login.this,MainActivity.class);
                startActivity(new Intent(getApplicationContext(),MainActivity
                .class));
                mintent.putExtra(VSEND, uname);
                startActivity(mintent);
            }else {
                Toast.makeText(Login.this, "Invalid Credentials!\n\tPlease try again",
                Toast.LENGTH_SHORT).show();
                progressBar.setVisibility(View.GONE);
            }
        });

    });

    // GoBack For Find Password
    mPassBtn.setOnClickListener
    (v -> startActivity(new Intent(getApplicationContext(),Register.class)));

    // No Internet Connection
    if(!isNetworkAvailable()==true) {
        new AlertDialog.Builder(this)
                .setIcon(android.R.drawable.ic_dialog_alert)
                .setTitle("Internet Connection Alert")
                .setMessage("Please Check Your Internet Connection")
                .setPositiveButton("Close", new DialogInterface
                .OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface,
                    int i) {
                        finish();
                    }
```

```
                }).show();
        }
        else if(isNetworkAvailable()==true) {
            Toast.makeText(Login.this,
                    "Welcome", Toast.LENGTH_LONG).show();
        }
    }

    public boolean isNetworkAvailable() {
        ConnectivityManager connectivityManager = (ConnectivityManager)
        getSystemService(Context.CONNECTIVITY_SERVICE);
        if (connectivityManager != null) {
            if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.Q) {
                NetworkCapabilities capabilities = connectivityManager
                .getNetworkCapabilities(connectivityManager.getActiveNetwork());
                if (capabilities != null) {
                    if (capabilities.hasTransport(NetworkCapabilities
                    .TRANSPORT_CELLULAR)) {

                        return true;
                    } else if (capabilities.hasTransport(NetworkCapabilities
                    .TRANSPORT_WIFI)) {

                        return true;
                    } else if (capabilities.hasTransport(NetworkCapabilities
                    .TRANSPORT_ETHERNET)) {

                        return true;
                    }
                }
            }
        }
        return false;
    }

    // Back Click For Exit
    @Override
    public void onBackPressed(){
        final AlertDialog.Builder builder = new AlertDialog.Builder(Login.this);
        builder.setMessage("Are you sure want to Exit ?");
        builder.setCancelable(true);
        builder.setNegativeButton("No", (dialogInterface, i)
        -> dialogInterface.cancel());
        builder.setPositiveButton("Yes", (dialogInterface, i) -> finish());
        AlertDialog alertDialog = builder.create();
```

```
        alertDialog.show();
    }
}
```

A.1.3 ForgetPass.java

```
package com.example.v2;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;

public class Register extends AppCompatActivity {

    private String email;

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

        EditText mEmail = findViewById(R.id.rEmail);
        TextView resetBtn = findViewById(R.id.resetBtn);
        TextView backBtn = findViewById(R.id.backBtn);

        FirebaseAuth fAuth = FirebaseAuth.getInstance();

        // Reset Password
        resetBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                validateData();
            }

            private void validateData() {
                email = mEmail.getText().toString().trim();
```

```
                if (email.isEmpty()) {
                    mEmail.setError("Required");
                }else {
                    forgotPass();
                }
            }

            private void forgotPass() {
                fAuth.sendPasswordResetEmail(email)
                        .addOnCompleteListener(task -> {
                            if (task.isSuccessful()){
                                Toast.makeText(Register.this,
                                "Check your email",Toast.LENGTH_SHORT).show();
                                startActivity(new Intent(getApplicationContext(),
                                Login.class)
                                .setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP));
                                finish();
                            }else {
                                Toast.makeText(Register.this,
                                "Please try again later",
                                Toast.LENGTH_SHORT).show();
                            }
                        });
            }
        });

        // Back to Login
        backBtn.setOnClickListener
        (v -> startActivity(new Intent(getApplicationContext(),Login.class)));
    }
}
```

A.1.4 SplashScreen.java

```
package com.example.v2;

import androidx.appcompat.app.AppCompatActivity;
import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;

@SuppressLint("CustomSplashScreen")
```

```
public class SplashScreen extends AppCompatActivity {

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

        new Handler().postDelayed(() -> {
            Intent i = new Intent(SplashScreen.this, Login.class);
            startActivity(i);
            finish();
        }, 3000);
    }
}
```

A.2 System Code

A.2.1 finalnotice.py

```
import time
import firebase_admin
import threading
from firebase_admin import credentials, firestore
from luma.led_matrix.device import max7219
from luma.core.interface.serial import spi, noop
from luma.core.render import canvas
from luma.core.virtual import viewport
from luma.core.legacy import text, show_message, textsize
from luma.core.legacy.font
import proportional, CP437_FONT, TINY_FONT, SINCLAIR_FONT, LCD_FONT
from gpiozero import Buzzer
from time import sleep
from datetime import datetime


serial = spi(port=0, device=0, gpio=noop())
device = max7219(serial,cascaded=16,block_orientation=-90
,blocks_arranged_in_reverse_order=True)

cred = credentials.Certificate("/home/pi/Downloads/adv.json")
firebase_admin.initialize_app(cred)
```

```
db = firestore.client()

# Create an Event for notifying main thread.
callback_done = threading.Event()

buzzer = Buzzer(17)

is_new_notice_available = False

old_notice_time = None

new_notice_data = {}

disp_notice_data = {}

def convert_to_date_string(date):
        if date is not None:
                return date.strftime('%d/%b/%Y')
        else: return "-"

def convert_to_time_string(date):
        if date is not None:
                return date.strftime('%I:%M:%S %p')
        else: return "-"

# Create a callback on_snapshot function to capture changes
def on_snapshot(doc_snapshot, changes, read_time):
    global old_notice_time
    global is_new_notice_available
    global new_notice_data

    for doc in doc_snapshot:
        timestamp = doc.get("DT")
        if old_notice_time is None:
            old_notice_time = timestamp

        if timestamp >= old_notice_time:
            timestamp_dt=datetime.fromtimestamp(doc.get("DT") / 1000)
            new_notice_data["text"] = doc.get('Text')
            new_notice_data["date"] = convert_to_date_string(timestamp_dt)
            new_notice_data["time"] = convert_to_time_string(timestamp_dt)


            uid=doc.get('uid')
            uid_doc = db.collection('users2').document(uid).get()
```

```
            new_notice_data["uname"] = uid_doc.get('uname')

            old_notice_time = timestamp

            is_new_notice_available = True

doc_ref = db.collection("users").order_by("DT",direction=firestore.Query
.DESCENDING).limit(1)
print("test2")

#Watch the document
doc_watch = doc_ref.on_snapshot(on_snapshot)

while(True):
    if new_notice_data.keys() != {"text", "date", "time", "uname"}:
        print("waiting for msg")
        continue

    if is_new_notice_available or disp_notice_data.keys()
    != {"text", "date", "time", "uname"}:
        print(is_new_notice_available)
        buzzer.on()
        time.sleep(1)
        buzzer.off()
        disp_notice_data = new_notice_data
        is_new_notice_available = False

    display_text = disp_notice_data["text"]
    display_date = disp_notice_data["date"]
    display_time = disp_notice_data["time"]
    display_uname = disp_notice_data["uname"]

    v1= f'Notice: {display_text} on {display_date} at {display_time}
    by {display_uname}'
    w, h = textsize(v1, font=proportional(CP437_FONT))
    print("Show running text: " + v1)
    show_message(device,v1 , fill="white", font=proportional(CP437_FONT)
    ,scroll_delay=0.04)
    time.sleep(0.5)
```

Bibliography

[1] Anushree, S., Bhat, D. V., Moonisha, G., and Venkatesh, U. (2014). Electronic notice board for professional college. *International Journal of Science, Engineering and Technology Research (IJSETR)*, 3(6).

[2] Arulmurugan, S., Anitha, S., Priyanga, A., and Sangeethapriya, S. (2016). Smart electronic notice board using wi-fi. *International Journal of Innovative Science, Engineering & Technology*, 3(3).

[3] Dalwadi, D. C., Trivedi, N., and Kasundra, A. (2011). Wireless notice board our real-time solution. In *National Conference on Recent Trends in Engineering & Technology*.

[4] Darekar, S., Davane, B., Khose, S., and Panigrahi, A. (2019). Real time notice display system using cloud.

[5] Dash, R. L. and Bevi, A. R. (2014). Real-time transmission of voice over 802.11 wireless networks using raspberry pi. *International Journal of Engineering Development and Research*, 2(1):2321–9939.

[6] Gaikwad, A., Kapadia, T., Lakhani, M., and Karia, D. (2013). Wireless electronic notice board. *nternational Journal on Advanced Computer Theory and Engineering (IJACTE)*, 2(2).

[7] Ghodeswar, D., Katare, A., Sudame, B., and Khojre, T. (2019). Voice based wireless notice board using gsm module.

[8] Jadhav, V. B., Nagwanshi, T. S., Patil, Y. P., and Patil, D. R. (2016). Digital notice board using raspberry pi. *International Research Journal of Engineering and Technology*, 3(5):2076–2079.

[9] Kadam, S., Saxena, A., and Gaurav, T. (2015). Android based wireless notice board and printer. *International Journal of Innovative Research in Computer and Communication Engineering*, 3(12).

[10] Ketkar, P. U., Tayade, K. P., Kulkarni, A. P., and Tugnayat, R. M. (2013). Gsm mobile phone based led scrolling message display system. *International Journal of Scientific Engineering and Technology*, 2(3):149–155.

[11] Kumar, P., Bhrdwaj, V., Pal, K., Rathor, N. S., and Mishra, A. (2012). Gsm based e-notice board: wireless communication. *International Journal of Soft Computing and Engineering*, 2(3):601–605.

[12] Merai, B., Jain, R., and Mishra, R. (2015). Smart notice board. *International Journal of Advanced Research in Computer and Communication Engineering*, 4(4):105–107.

[13] Mujumdar, A., Niranjane, V., and Sagne, D. (2014). Scrolling led display using wireless transmission.

[14] Priyadharsini, M., Arunbalaji, V., and Karthikaa, T. (2016). Gsm based motor control for irrigation system. In *National Conference on Recent Trends in Electronics and Instrumentation Engineering (NCRTE 2K16)*, volume 1, page 2nd.

[15] Sharma, R., Singh, P., and Singhal, A. (2017). Two different authentication protocol for rfid credit card security. *Journal of Engineering and Applied Sciences*, 12(22):6006–6012.

[16] Yakaiah, P., Bijjam Swathi, M. J., Nikhala, B., and Prasad, K. S. (2017). Remotely controlled android based electronic notice board. *International Journal of Scientific Development and Research (IJSDR)*, 2(4):2455–2631.

[17] Yinan, G., Shuguo, Z., and Dawei, X. (2012). Overview of wi-fi technology. In *The 2nd International Conference on Computer Application and System Modeling.*