3 Factor Authentication using RFID based Access Control

Dhruvit Nimavat
Electronics and Communication Engineering
Institute of Technology, Nirma University
Ahmedabad, Gujarat
21bec075@nirmauni.ac.in

Krinal Parmar
Electronics and Communication Engineering
Institute of Technology, Nirma University
Ahmedabad, Gujarat
21bec084@nirmauni.ac.in

***Abstract*—This paper presents a design scheme that combines embedded technology with RFID based on Raspberry Pi. This system provides 3 factor authentication using RFID tags and scanners, PIN and SMS. From the test, the system demonstrates the implementation of authentication function with many features including easy-to-operate, fast identification, and high data security. Raspberry Pi OS, based on Debian Bullseye, is an open-source operating system tailored for the Raspberry Pi single-board computer which is used to implement the authentication access control system. It is optimized for offering improved functionality and compatibility with peripherals like keyboards, mice, and displays. We achieved the 3 factor authentication using Raspberry Pi alongwith the 3 factors, which are RFID tags and readers, PIN and SMS, and lay out the access providing system with security.**

Keywords—RFID, Raspberry Pi, Access Control, Authentication, PIN, SMS

I. Introduction

Authentication system is widely used for residential security, office buildings, data centres, educational institutions, healthcare facilities. Here, this system is used for door lock system which offers a versatile solution for securing physical spaces across various sectors, providing robust access control and enhancing overall security measures in day-to-day life applications. For instance, it is used for biometric recognition and other scenarios[1]. RFID card makes use of space electromagnetic propagation to communicate, in order to achieve non-contact automatic identification purpose of wireless communication technology. Attempts to provide complex security system exist, but they do not cover whole spectrum of digital system.

The next section contains a brief review of a typical RFID technology to increase the security and provide authentication. Also, discussion about Raspberry Pi being the main hardware component is provided. The model block diagram[6] and the detailed explanation of the system hardware design implemented is been discussed in the next section.

II. RFID (Radio Frequency Identification)

Radio Frequency Identification also known as RFID is known for the wireless communication possible using radio frequency tracking and identification of objects. RFID system consist of RFID tags, RFID reader and a system to perform the task of tracking. Firstly, they were used for military purpose to track enemy and measure the distance of their planes[2]. Nowadays, they are modified and are used for various purposes.

A. *Kinds of RFID*

RFID tags can be of two kind, namely, passive and active. Passive RFID tags neither have a battery nor use internal power source. They rely on energy from the RFID reader and its antenna power. Active RFID tags has its own power supply and are continuously operating used to track objects within the range.

B. *Working principle of RFID*

- RFID requires RFID tags and RFID reader alongwith a system to track the object. An RFID reader scans the tag and data is provided in the system through it.
- The data is transmitted to the database, where it is being processed and stored.
- The tags and readers identify the electronic ID and the system manages the corresponding information as per the requirement.

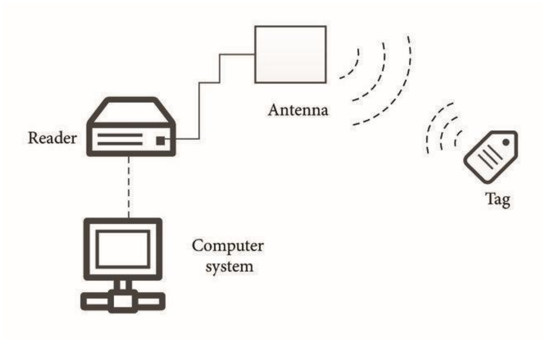


Fig. 1. Working principle of RFID system

C. *Features of RFID*

- RFID provides means to accurately identify the objects individually and the system used the ID to update the records with the combination of various data.
- Automatic recognition of ID to the readers is achieved. The readers are located such that it scans the ID and recognizes the object.
- Passive RFID tags do not require batteries and rely on the energy transmitted by the reader. But for longer distance, active tags are required as passive tags work within a certain range.

III. Raspberry Pi

Raspberry Pi is a series of small single board computers developed in UK by Raspberry Foundation. It can be used as mini personal computer by connecting peripherals like keyboards, mouse, display to the Raspberry Pi. It is known for its use in real time Image and Video Processing, IOT based applications and Robotics applications.

Fig. 2. Raspberry Pi

A. *Raspberry Pi 3 Model B+*

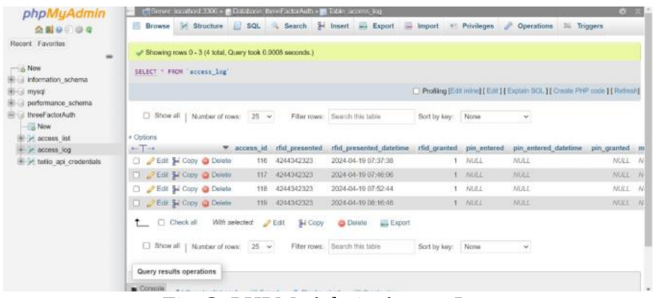
Raspberry Pi 3 Model B+ has a 64-bit quad-core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, Gigabit Ethernet over USB 2.0, and PoE capability via a separate PoE HAT. The dual-band wireless LAN comes with modular compliance certification. Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both Raspberry Pi 2 Model B and Raspberry Pi 3 Model B

B. *Raspberry Pi Operating System*

The Raspberry Pi OS is a free, open-source operating system based on the Debian operating system. It is designed specifically for the Raspberry Pi single-board computer, and is optimized for use with peripherals such as keyboards, mice, and displays. The latest version of Raspberry Pi OS, as of July 2021, is Debian Bullseye. This version includes updates and improvements over the previous version, and is recommended for those looking to use the Raspberry Pi for both educational and hobby projects.

IV. PHPMYADMIN

PHPMyAdmin is a versatile web-based application designed for managing MySQL and MariaDB databases through a user-friendly interface. With PHPMyAdmin, users can execute SQL queries effortlessly, enabling them to interact with their databases efficiently. Its intuitive design simplifies tasks such as creating, modifying, and deleting database structures, as well as importing and exporting data. Additionally, PHPMyAdmin offers features for user management, server monitoring, and SQL query optimization, making it an indispensable tool for database administrators, developers, and anyone else working with MySQL or MariaDB databases. Whether for simple data retrieval or complex database management tasks, PHPMyAdmin provides a convenient and accessible solution for SQL database management.



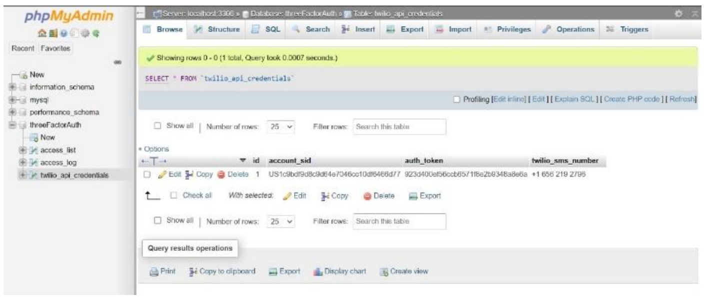
Showing rows 0 - 3 (4 total, Query took 0.0008 seconds.)

```
SELECT * FROM `access_log`
```

access_id	rfid_presented	rfid_presented_datetime	rfid_granted	pin_entered	pin_entered_datetime	pin_granted
116	4244342323	2024-04-19 07:37:38	1	NULL	NULL	NULL
117	4244342323	2024-04-19 07:46:06	1	NULL	NULL	NULL
118	4244342323	2024-04-19 07:52:44	1	NULL	NULL	NULL
119	4244342323	2024-04-19 08:16:46	1	NULL	NULL	NULL

Fig. 3. PHPMyAdmin Access Log

Our system utilizes a robust database infrastructure to securely store user information, granting access exclusively to authorized individuals. To bolster security measures, we maintain comprehensive logs detailing all access attempts, including unsuccessful ones, serving as a vital security feature. Additionally, we utilize this platform to securely store sensitive credentials, such as those required for Twilio API integration, ensuring the integrity of our data and operations.



Showing rows 0 - 0 (1 total, Query took 0.0007 seconds.)

```
SELECT * FROM `twilio_api_credentials`
```

id	account_sid	auth_token	twilio_sms_number
1	US1c9bdf9d8c9d64e7046cc10df6466d77	923d400ef56ccb6571f8e2b9348a8e6a	+1 656 219 2795

Fig. 4. PHPMyAdmin Twilio Credentials

V. ACCESS BASED CONTROL SYSTEM

The access-based control system presents a comprehensive security solution leveraging RFID authentication, PIN entry, and SMS verification. By integrating these multiple layers if authentication, the system ensures enhanced security, ranging from residential to commercial settings. The GUI simplifies user interaction, enhancing its usability and accessibility[3].

A. *Functionality*

Aspects	**3-Factor Authentication**	**2-Factor Authentication**
Number of Factors	Requires authentication based on three factors: knowledge factor, possession factor and inherence factor.	Requires authentication based on two factors: knowledge factor and possession factor.
Security Strength	Offers a higher level of security due to combination of multiple factors making it harder for unauthorized access.	Provides moderate level of security by combining two factors for authentication but may be less robust than three-way authentication.
Implementation complexity	More complex to implement and manage due to the integration of multiple factors and associated system.	Simpler to implement and manage compared to three-way authentication, as it involves few factors and systems.
User Experience	May introduce more steps in authentication process, potentially requiring additional actions such as biometric scans and additional hardware.	Generally, offers a smoother user experience as it typically involves fewer steps.
Use Cases	Commomly used in high security environments such as government agencies, financial institutions and highly regulated industries where stringent security measures are required.	Widely adopted across various sectors, including banking, online services, and mobile applications, due to its balance between security and usability.

B. *Algorithm*

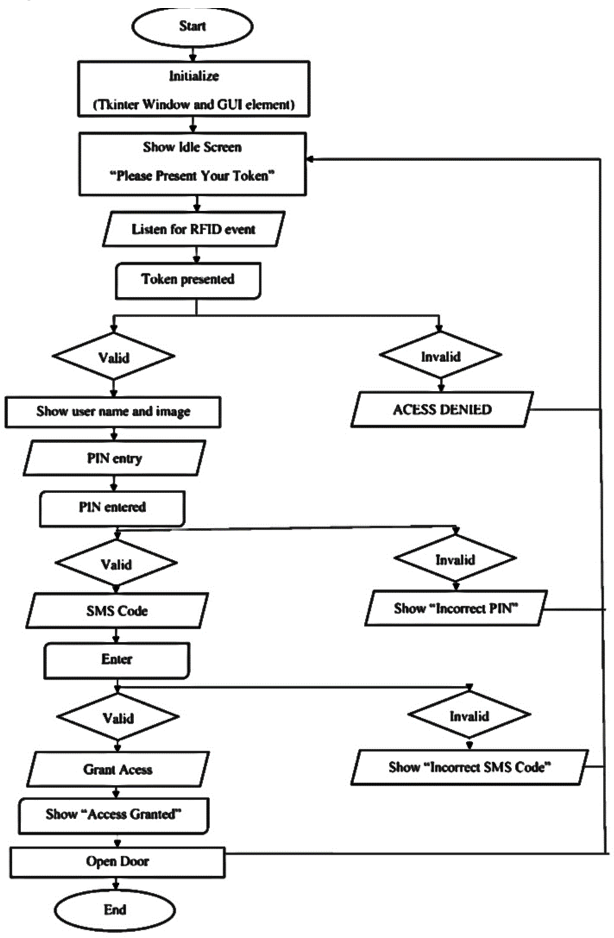


Fig. 5. Flowchart of the system

C. *System Hardware Design*

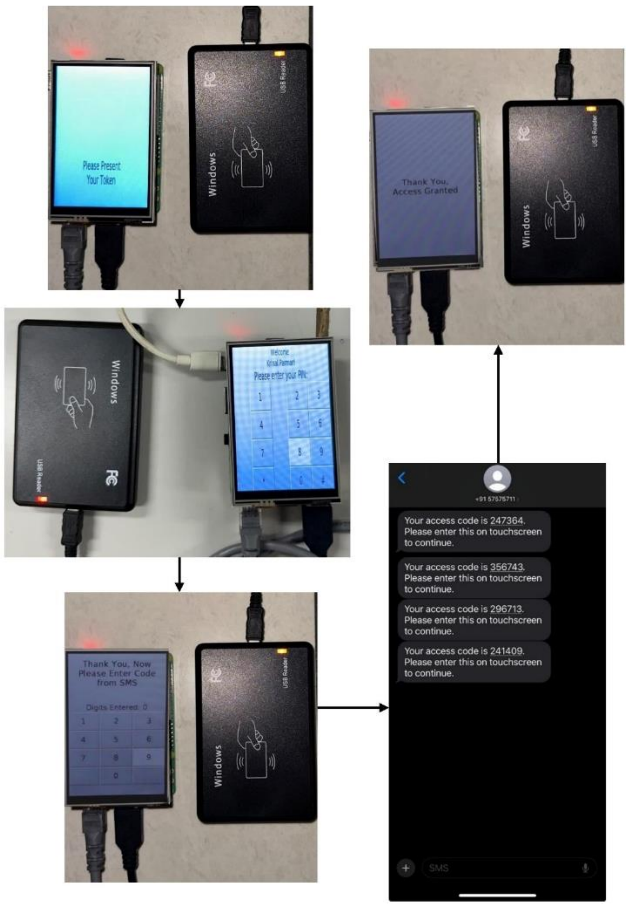
The access based control system works in multiple layers divided into three authentication, namely, RFID tag, PIN and SMS. Here given below is the explanation of each layer:

- RFID Authentication: When a user presents an RFID token, the system queries a database to validate its authenticity. If the token is valid, the system proceeds to the next step.
- PIN Entry: The user is prompted to enter a PIN via the GUI. Upon PIN entry, the system verifies the correctness of the PIN against the user's stored credentials. If the PIN is correct, the system sends an SMS code to the user's registered mobile number for further verification.
- SMS Verification: A SMS code is sent to the user's device which the user have to enter into the system, and the system checks the correctness of the code before accessing the grant to the user.
- Access Control: Once authenticated, the system automatically unlocks the door as per the access being granted on passing the authentication and lets the user physically in[5]. Also additionally, the access attempts and results are recorded for tracking and auditing.
- Feedback and Timeout Handling: The system updates the user's status via the GUI throughout

the process. It includes the timeout mechanism to return to idle after a certain period of time for security and the ease of use.

By combining RFID authentication, PIN entry[7], and SMS verification, the system employs a multi-factor

authentication approach to enhance security. This working principle ensures that only authorized users with valid credentials and mobile devices can gain access to the controlled area.


Please Present Your Token
Windows
USB Reader
Welcome
Please enter your PIN
Thank You, Now Please Enter Code from SMS
Digits Entered: 2
Thank You, Access Granted
+91 57575711
SMS

Your access code is 247364. Please enter this on touchscreen to continue.

Your access code is 356743. Please enter this on touchscreen to continue.

Your access code is 296713. Please enter this on touchscreen to continue.

Your access code is 241409. Please enter this on touchscreen to continue.

Fig. 6. Hardware Implementation

D. System Software Design

A powerful security programme was created especially for door lock systems that use Raspberry Pi. It guarantees improved access control by using three-factor authentication. This software offers numerous layers of protection to protect access to critical regions by seamlessly integrating with PIN entry, SMS verification, and RFID card authentication. It is a user-friendly interface that maintains strong security levels, thanks to its development in Python and Tkinter. Data integrity and confidentiality are guaranteed by the system's safe storage of user information, access logs, and Twilio API credentials in a MySQL database.

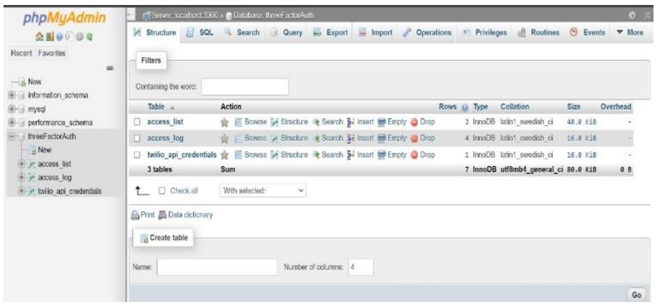
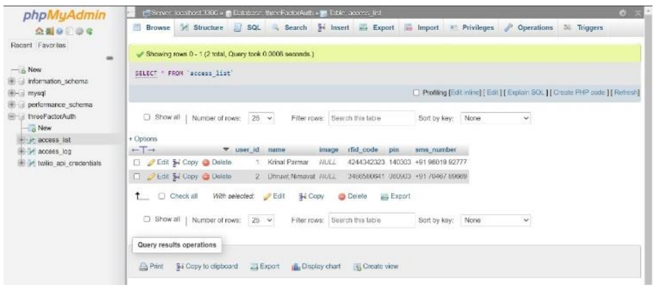


Table	Action	Rows	Type	Collation	Size	Overhead
access_list	Browse Structure Search Insert Empty Drop	2	InnoDB	latin1_swedish_ci	48.0 KiB	-
access_log	Browse Structure Search Insert Empty Drop	4	InnoDB	latin1_swedish_ci	16.0 KiB	-
twilio_api_credentials	Browse Structure Search Insert Empty Drop	1	InnoDB	latin1_swedish_ci	16.0 KiB	-
3 tables	Sum	7	InnoDB	utf8mb4_general_ci	80.0 KiB	0 B

Fig. 7. PHPMyAdmin SQL

By reading this programme description, users will obtain knowledge about it's architecture, implementation, and functionality, enabling them to set up and modify cutting-edge security solutions for their Internet of Things applications.



Showing rows 0 - 1 (2 total, Query took 0.0006 seconds.)

```
SELECT * FROM `access_list`
```

	user_id	name	image	rfid_code	pin	sms_number
Edit Copy Delete	1	Krinal Parmar	NULL	4244342323	140303	+91 98019 92777
Edit Copy Delete	2	Dhruvit Nimavat	NULL	3466580641	080903	+91 70467 89869

Fig. PHPMyAdmin Access User

- Features:
  - ⇒ Three-factor authentication: RFID card, PIN entry, and SMS verification.
  - ⇒ Python and Tkinter-based user interface for easy interaction.
  - ⇒ Integration with Twilio API for secure SMS authentication.
  - ⇒ Secure storage of user data and access logs in a MySQL database.
  - ⇒ Flexible and customizable architecture for diverse security requirements.
  - ⇒ Scalable design suitable for small-scale to enterprise-level deployments.
- Key Components:
  - ⇒ RFID Card Authentication: SecurePi authenticates users via RFID cards, ensuring swift and convenient access.
  - ⇒ PIN Entry: Users are required to enter a PIN for additional authentication, enhancing security.
  - ⇒ SMS Verification: SecurePi sends SMS codes to registered mobile numbers for verification, adding an extra layer of security.
  - ⇒ Database Management: Utilizes MySQL database for secure storage of user information, access logs, and API credentials.
  - ⇒ Twilio Integration: Integrates with Twilio's API to facilitate secure SMS communication for authentication purposes.

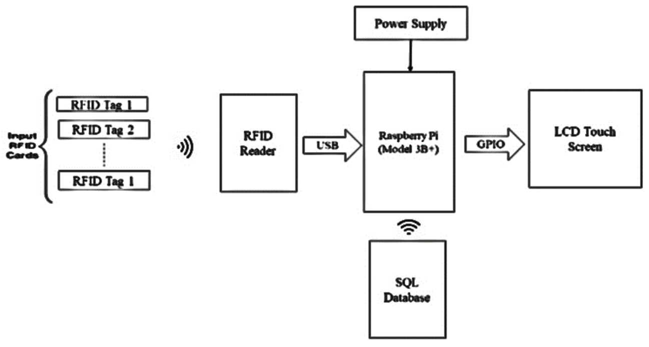


Fig. 5. Block Diagram

VI. Conclusion

Using RFID technology in authentication allows to increase their security. Even low-end RFID tags can add one more security level when combined with physical access control systems. Intellectual RFID tags with possibility of strong mutual authentication with smart cards allow to provide unauthorized access to a system: they can be used after successful mutual authentication only.

In this paper we implemented typical 3 factor authentication system and commonly used mechanisms to obtain their security. We proposed a way to combine RFID-based physical access control systems with PIN and SMS to increase their security that allows to prevent unauthorized access.

References

[1] G. J. Dharmale, J. Katti, S. Waghere, T. Patankar and K. Ati, "Door Lock using RFID and Arduino," *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Kharagpur, India, 2022, pp. 1-5, doi: 10.1109/ICCCNT54827.2022.9984396.

[2] J. W. Simatupang and R. W. Tambunan, "Security Door Lock Using Multi-Sensor System Based on RFID, Fingerprint, and Keypad," *2022 International Conference on Green Energy, Computing and Sustainable Technology (GECOST)*, Miri Sarawak, Malaysia, 2022, pp. 453-457, doi: 10.1109/GECOST55694.2022.10010367.

[3] M. Mathew and R. S. Divya, "Super secure door lock system for critical zones," *2017 International Conference on Networks & Advances in Computational Technologies (NetACT)*, Thiruvananthapuram, India, 2017, pp. 242-245, doi: 10.1109/NETACT.2017.8076773.

[4] R. R. Gangi and S. S. Gollapudi, "Locker opening and vlosing system using RFID fingerprint password and GSM", Int. J. Emerg. Trends Technol. Comput. Sci. (IJETTCS), vol. 2, no. 2, pp. 142-145, Mar.–Apr. 2013, [online] Available: http://www.ijettcs.org/Volume2Issue2/IJETTCS-2013-04-03-060.pdf.

[5] U. R. El-Yakub, K. Jimpa, A. K. Mandal and H. Saha, "Blynk, Database and WhatsApp Integrated RFID Based Lock System," *2023 26th International Conference on Computer and Information Technology (ICCIT)*, Cox's Bazar, Bangladesh, 2023, pp. 1-6, doi: 10.1109/ICCIT60459.2023.10441308.

[6] K. Rudraraju, S. G. L. Divakarla and J. N. V. Vardhan, "Door Locking System using RFID and GSM Technology," *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, 2023, pp. 379-381.

[7] A. Ashraf, D. Rasaily and A. Dahal, "Password protected lock system designed using microcontroller", Int. J. Eng. Trends Technol. (IJETT), vol. 32, no. 4, pp. 180-183, Feb. 2016, [online] Available: https://www.ijcaonline.org/archives/volume153/number2/nehete-2016-ijca-911971.pdf.

Appendix-I

A. Python Script:

```
#!/usr/bin/env python3
import sys
import MySQLdb
from threading import Thread
import threading
import time
import RPi.GPIO as GPIO
import json
from random import randint
from evdev import InputDevice
from select import select
from twilio.rest import Client

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(13,GPIO.OUT)

try:
        # python 2
        import Tkinter as tk
        import ttk
```

```
except ImportError:
	# python 3
	import tkinter as tk
	from tkinter import ttk

class Fullscreen_Window:

	global dbHost
	global dbName
	global dbUser
	global dbPass

	dbHost = 'localhost'
	dbName = 'threeFactorAuth'
	dbUser = 'phpmyadmin'
	dbPass = 'raspberry'

	def _init_(self):
		self.tk = tk.Tk()
		self.tk.title("Three-Factor    Authentication
Security Door Lock")
		self.frame = tk.Frame(self.tk)
		self.frame.grid()
		self.tk.columnconfigure(0, weight=1)

		self.tk.attributes('-zoomed', True)
		self.tk.attributes('-fullscreen', True)
		self.state = True
		self.tk.bind("<F11>",
self.toggle_fullscreen)
		self.tk.bind("<Escape>",
self.end_fullscreen)
		self.tk.config(cursor="none")

		self.show_idle()

		t = Thread(target=self.listen_rfid)
		t.daemon = True
		t.start()

	def show_idle(self):
		self.welcomeLabel    =    ttk.Label(self.tk,
text="Please Present\nYour Token")
		self.welcomeLabel.config(font='size,    20',
justify='center', anchor='center')
		self.welcomeLabel.grid(sticky=tk.W+tk.E,
pady=210)

	def pin_entry_forget(self):
		self.validUser.grid_forget()
		self.photoLabel.grid_forget()
		self.enterPINlabel.grid_forget()
		count = 0
		while (count < 12):
			self.btn[count].grid_forget()
			count += 1

	def returnToIdle_fromPINentry(self):
		self.pin_entry_forget()
		self.show_idle()

	def returnToIdle_fromPINentered(self):
		self.PINresultLabel.grid_forget()
		self.show_idle()

	def returnToIdle_fromAccessGranted(self):
		GPIO.output(13,GPIO.LOW)
		self.SMSresultLabel.grid_forget()
		self.show_idle()

	def returnToIdle_fromSMSentry(self):
		self.PINresultLabel.grid_forget()
		self.smsDigitsLabel.grid_forget()
		count = 0
		while (count < 12):
			self.btn[count].grid_forget()
			count += 1
		self.show_idle()

	def	returnToIdle_fromSMSentered(self):
		self.SMSresultLabel.grid_forget()
		self.show_idle()

	def toggle_fullscreen(self, event=None):
		self.state = not self.state   # Just toggling
the boolean
		self.tk.attributes("-fullscreen", self.state)
		return "break"

	def end_fullscreen(self, event=None):
		self.state = False
		self.tk.attributes("-fullscreen", False)
		return "break"

	def listen_rfid(self):
		global pin
		global accessLogId

		keys                                        =
"X^1234567890XXXXqwertzuiopXXXXasdfghjklXXXXXyxcv
bnmXXXXXXXXXXXXXXXXXXXXXXX"
		dev = InputDevice('/dev/input/event0')
		rfid_presented = ""

		while True:
			r,w,x = select([dev], [], [])
			for event in dev.read():
				if    event.type==1    and
event.value==1:
								if
event.code==28:

	dbConnection   =   MySQLdb.connect(host=dbHost,
user=dbUser, passwd=dbPass, db=dbName)

	cur                                         =
dbConnection.cursor(MySQLdb.cursors.DictCursor)

	cur.execute("SELECT * FROM access_list WHERE
rfid_code = '%s'" % (rfid_presented))



	if cur.rowcount != 1:
```

```
                self.welcomeLabel.config(text="ACCESS DENIED")


                # Log access attempt

                cur.execute("INSERT INTO access_log SET rfid_presented = '%s', rfid_presented_datetime = NOW(), rfid_granted = 0" % (rfid_presented))

                dbConnection.commit()


                time.sleep(3)

                self.welcomeLabel.grid_forget()

                self.show_idle()

        else:

                user_info = cur.fetchone()

                userPin = user_info['pin']

                self.welcomeLabel.grid_forget()

                self.validUser = ttk.Label(self.tk, text="Welcome\n %s!" % (user_info['name']), font='size, 15', justify='center', anchor='center')

                self.validUser.grid(columnspan=3, sticky=tk.W+tk.E)


                self.image = tk.PhotoImage(file=user_info['image'] + ".gif")

                self.photoLabel = ttk.Label(self.tk, image=self.image)

                self.photoLabel.grid(columnspan=3)


                self.enterPINlabel = ttk.Label(self.tk, text="Please enter your PIN:", font='size, 18', justify='center', anchor='center')

                self.enterPINlabel.grid(columnspan=3, sticky=tk.W+tk.E)

                pin = ''


                keypad = [

                        '1', '2', '3',

                        '4', '5', '6',

                        '7', '8', '9',

                        '*', '0', '#',

                ]


                # create and position all buttons with a for-loop

                # r, c used for row, column grid values

                r = 4

                c = 0

                n = 0

                # list(range()) needed for Python3

                self.btn = list(range(len(keypad)))

                for label in keypad:

                        # partial takes care of function and argument

                        #cmd = partial(click, label)

                        # create the button

                        self.btn[n] = tk.Button(self.tk, text=label, font='size, 18', width=4, height=1, command=lambda digitPressed=label:self.codeInput(digitPressed, userPin, user_info['sms_number']))

                        # position the button

                        self.btn[n].grid(row=r, column=c, ipadx=10, ipady=10)

                        # increment button index

                        n += 1

                        # update row/column position

                        c += 1

                        if c > 2:

                                c = 0

                                r += 1
```

```
					# Log access attempt

					cur.execute("INSERT INTO access_log SET rfid_presented = '%s', rfid_presented_datetime = NOW(), rfid_granted = 1" % (rfid_presented))

					dbConnection.commit()

					accessLogId = cur.lastrowid



					self.PINentrytimeout = threading.Timer(10, self.returnToIdle_fromPINentry)

					self.PINentrytimeout.start()



					self.PINenteredtimeout = threading.Timer(5, self.returnToIdle_fromPINentered)



				rfid_presented = ""

				dbConnection.close()
						else:

				rfid_presented += keys[ event.code ]

	def codeInput(self, value, userPin, mobileNumber):
		global accessLogId
		global pin
		global smsCodeEntered
		pin += value
		pinLength = len(pin)

		self.enterPINlabel.config(text="Digits Entered: %d" % pinLength)

		if pinLength == 6:
			self.PINentrytimeout.cancel()
			self.pin_entry_forget()

			if pin == userPin:
				pin_granted = 1
			else:
				pin_granted = 0

			# Log access attempt
			dbConnection = MySQLdb.connect(host=dbHost, user=dbUser, passwd=dbPass, db=dbName)
			cur = dbConnection.cursor()
			cur.execute("UPDATE access_log SET pin_entered = '%s', pin_entered_datetime = NOW(), pin_granted = %s, mobile_number = '%s' WHERE access_id = %s" % (pin, pin_granted, mobileNumber, accessLogId))
			dbConnection.commit()

			if pin == userPin:
				self.PINresultLabel = ttk.Label(self.tk, text="Thank You, Now\nPlease Enter Code\nfrom SMS\n")

		self.PINresultLabel.config(font='size, 20', justify='center', anchor='center')

		self.PINresultLabel.grid(columnspan=3, sticky=tk.W+tk.E, pady=20)

				self.smsDigitsLabel = ttk.Label(self.tk, text="Digits Entered: 0", font='size, 18', justify='center', anchor='center')

		self.smsDigitsLabel.grid(columnspan=3, sticky=tk.W+tk.E)

				smsCode = self.sendSMScode(mobileNumber)
				smsCodeEntered = ''

				keypad = [
					'1', '2', '3',
					'4', '5', '6',
					'7', '8', '9',
					'', '0', '',
				]

				# create and position all buttons with a for-loop
				# r, c used for row, column grid values
				r = 4
				c = 0
				n = 0
				# list(range()) needed for Python3
				self.btn = list(range(len(keypad)))
				for label in keypad:
					# partial takes care of function and argument
					#cmd = partial(click, label)
					# create the button
					self.btn[n] = tk.Button(self.tk, text=label, font='size, 18', width=4, height=1, command=lambda digitPressed=label:self.smsCodeEnteredInput(digitPressed, smsCode))
					# position the button

		self.btn[n].grid(row=r, column=c, ipadx=10, ipady=10)
```

```
                    # increment button index
                    n += 1
                    # update row/column position
                    c += 1
                    if c > 2:
                        c = 0
                        r += 1

                self.SMSentrytimeout = threading.Timer(60, self.returnToIdle_fromSMSentry)
    self.SMSentrytimeout.start()

            else:
                self.PINresultLabel = ttk.Label(self.tk, text="Incorrect PIN\nEntered!")
    self.PINresultLabel.config(font='size, 20', justify='center', anchor='center')

    self.PINresultLabel.grid(sticky=tk.W+tk.E, pady=210)

    self.PINenteredtimeout.start()

    def smsCodeEnteredInput(self, value, smsCode):
        global smsCodeEntered
        global accessLogId
        smsCodeEntered += value
        smsCodeEnteredLength = len(smsCodeEntered)

        self.smsDigitsLabel.config(text="Digits Entered: %d" % smsCodeEnteredLength)

        if smsCodeEnteredLength == 6:
            self.SMSentrytimeout.cancel()
            self.pin_entry_forget()

            if smsCodeEntered == smsCode:
                smscode_granted = 1
            else:
                smscode_granted = 0

            # Log access attempt
            dbConnection = MySQLdb.connect(host=dbHost, user=dbUser, passwd=dbPass, db=dbName)
            cur = dbConnection.cursor()
            cur.execute("UPDATE access_log SET smscode_entered = '%s', smscode_entered_datetime = NOW(), smscode_granted = %s WHERE access_id = %s" % (smsCodeEntered, smscode_granted, accessLogId))
            dbConnection.commit()

            if smsCodeEntered == smsCode:
                self.SMSresultLabel = ttk.Label(self.tk, text="Thank You,\nAccess Granted")

    self.SMSresultLabel.config(font='size, 20', justify='center', anchor='center')

    self.SMSresultLabel.grid(columnspan=3, sticky=tk.W+tk.E, pady=210)


    self.PINresultLabel.grid_forget()

    self.smsDigitsLabel.grid_forget()

    GPIO.output(13,GPIO.HIGH)

                self.doorOpenTimeout = threading.Timer(10, self.returnToIdle_fromAccessGranted)

    self.doorOpenTimeout.start()
            else:

    self.PINresultLabel.grid_forget()

    self.smsDigitsLabel.grid_forget()

                self.SMSresultLabel = ttk.Label(self.tk, text="Incorrect SMS\nCode Entered!")

    self.SMSresultLabel.config(font='size, 20', justify='center', anchor='center')

    self.SMSresultLabel.grid(sticky=tk.W+tk.E, pady=210)

                self.SMSenteredtimeout = threading.Timer(10, self.returnToIdle_fromSMSentered)

    self.SMSenteredtimeout.start()

    def sendSMScode(self, mobileNumber):

        # Retreive our Twilio access credentials and "from" number
        dbConnection = MySQLdb.connect(host=dbHost, user=dbUser, passwd=dbPass, db=dbName)
        cur = dbConnection.cursor(MySQLdb.cursors.DictCursor)
        cur.execute("SELECT account_sid, auth_token, twilio_sms_number FROM twilio_api_credentials WHERE id = 1")
        credentials = cur.fetchone()
        account_sid = credentials['account_sid']
        auth_token = credentials['auth_token']
        twilio_sms_number = credentials['twilio_sms_number']
        dbConnection.close()

        smsCode = str(randint(100000, 999999))
        messageText = "Your access code is %s. Please enter this on the touchscreen to continue." % smsCode

        client = Client(account_sid, auth_token)
```

```
            message = client.messages.create(
                to=mobileNumber,
                from_=twilio_sms_number,
                body=messageText)

            return smsCode

if _name_ == '_main_':
    w = Fullscreen_Window()
    w.tk.mainloop()
```

B. SQL Database:

```
phpMyAdmin SQL Dump
-- version 5.0.4deb2+deb11u1
-- https://www.phpmyadmin.net/
--
-- Host: localhost:3306
-- Generation Time: Apr 18, 2024 at 10:27 PM
-- Server version: 10.5.23-MariaDB-0+deb11u1
-- PHP Version: 7.4.33

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: threeFactorAuth
--

-- --------------------------------------------------------

--
-- Table structure for table access_list
--

CREATE TABLE access_list (
  user_id bigint(10) NOT NULL,
  name varchar(100) NOT NULL,
  image varchar(50) DEFAULT NULL,
  rfid_code varchar(20) NOT NULL,
  pin char(6) NOT NULL,
  sms_number varchar(15) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;


--
-- Dumping data for table access_list
--

INSERT INTO access_list (user_id, name, image, rfid_code, pin, sms_number) VALUES
(1, 'Krinal Parmar', 'Krinal', '4244342323', '140303', '+91 96019 92777'),
(2, 'Dhruvit Nimavat', 'Dhruvit', '3486580641', '060903', '+91 70467 89669');

-- --------------------------------------------------------

--
-- Table structure for table access_log
--

CREATE TABLE access_log (
  access_id int(11) NOT NULL,
  rfid_presented varchar(20) DEFAULT NULL,
  rfid_presented_datetime datetime DEFAULT NULL,
  rfid_granted tinyint(1) DEFAULT NULL,
  pin_entered char(6) DEFAULT NULL,
  pin_entered_datetime datetime DEFAULT NULL,
  pin_granted tinyint(1) DEFAULT NULL,
  mobile_number varchar(15) DEFAULT NULL,
  smscode_entered char(6) DEFAULT NULL,
  smscode_entered_datetime datetime DEFAULT NULL,
  smscode_granted tinyint(1) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

-- --------------------------------------------------------

--
```

```
-- Table structure for table twilio_api_credentials
--

CREATE TABLE twilio_api_credentials (
  id int(11) NOT NULL,
  account_sid varchar(50) NOT NULL,
  auth_token varchar(50) NOT NULL,
  twilio_sms_number varchar(15) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_swedish_ci;

--
-- Dumping data for table twilio_api_credentials
--

INSERT INTO twilio_api_credentials (id, account_sid, auth_token, twilio_sms_number) VALUES
(1, 'US1c9bdf9d8c9d64e7046cc10df6466d77', '923d400ef56ccb6571f8e2b9348a8e6a', '+1 656 219 2796');

--
-- Indexes for dumped tables
--

--
-- Indexes for table access_list
--
ALTER TABLE access_list
  ADD PRIMARY KEY (user_id),
  ADD UNIQUE KEY rfid_code (rfid_code),
  ADD UNIQUE KEY image (image);

--
-- Indexes for table access_log
--
ALTER TABLE access_log
  ADD PRIMARY KEY (access_id);

--
-- Indexes for table twilio_api_credentials
--
ALTER TABLE twilio_api_credentials
  ADD PRIMARY KEY (id);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table access_list
--
ALTER TABLE access_list
  MODIFY user_id bigint(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT for table access_log
--
ALTER TABLE access_log
  MODIFY access_id int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=116;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```