Smart Safe Lock

Desu,Sai Teja
Dept: Software Engineering
saiteja.desu@sjsu.edu
Class ID: 7

Bhavirisetty,Akhil
Dept: Software Engineering
akhil.bhavirisetty@sjsu.edu
Class ID: 3

Singh,Vinitkumar Ramprakash
Dept: Software Engineering
vinitkumarramprakash.singh@sjsu.edu
Class ID: 36

***Abstract (Bhavirisetty, Akhil)*— The importance of security in safe locks has been gaining a lot these days since most of the current digital safe locks are breakable by using a simple software. In addition, the current safe locks require either carrying a physical key or remembering the passcode, which are prone to lose the key or forget the password. The purpose of our project is to develop a smart system which unlocks the safe through a voice command and captures the intruder's photo who tries to open the safe. Moreover, the smart safe lock can alert the owners through email and text messaging system.**

Keywords—Safe Lock, Digital Safe Lock, Chatbot, Intruder Tracking

I. INTRODUCTION (DESU,SAI TEJA)

Safe locks are becoming prominent now a days to store valuable objects such as jewelry, cash, and important documents. There are different types of safe locks available in the market, which are mechanical combination locks, digital combination locks, and physical key locks. Most of the locks are vulnerable and hence not safe. Also, using current locks in the market is quite challenging in terms of keeping the physical key safe or remembering the digital passcode all the time. Digital passcodes are better than carrying a key, however it is difficult to remember the secret code for elder people. Security of safe locks is very important to protect crucial things in the safe, which existing systems lack. The Smart Lock Safe application addresses all the challenges and loop holes in present safe lock systems.

Smart lock safe is an alternative and secure application to the digital lock safe. An intelligent chatbot is associated with the safe to interact with the owner to unlock the safe. The chatbot is designed to understand the natural language (English) for receiving the One Time Password (OTP) from humans and respond in case of a wrong attempt through speech. The proposed application also consists of camera, which takes the snapshot of burglars. In addition, the smart lock safe will provide notification to the owner by sending text messages and email in case of a suspicious activities.

The purpose of this report is to provide comprehensive details such as implementation, features, technologies, performance testing, challenges, and future work of the developed application.

II. APPLICATION OVERVIEW (*SINGH, VINITKUMAR RAMPRAKASH*)

The smart safe lock application is designed to be easy to use and portable. There are many unique features defined for the application explained in following sections.

A. Features

- Unlocking through dynamic OTP
- Interactive voice chat
- Intruder alerts
- Captures photos of burglars
- Email and Text messaging notification
- Keyless
- Independent of digital passcodes

B. Requirements

Developing smart lock safe requires both software and hardware. Below are the key requirements for this application.

Hardware Requirements:

- Raspberry Pi 2 or above version
- USB Microphone
- USB Speaker
- Pi Camera
- Ethernet Connecting cable
- Wi-Fi Router
- Micro SD Card (> 8GB)

Software Requirements:

- Latest Raspbian OS
- Latest python
- Speech recognition libraries
- Configure SMTP Protocol
- REST API

III. SYSTEM DESIGN & IMPLEMENTATION (DESU,SAI TEJA)

A. Components

Raspberry Pi 2: The Raspberry pi is the smallest powerful computer which can perform any task same as like normal personal computer. The Pi is very less expensive and thus makes any person of any age to explore computer. Raspberry pi is widely used in Internet of Things due to low cost and it's portable nature. Raspberry Pi is the core component of our application, which processes all the requests from the user and runs corresponding python script, then responds to user with appropriate answers.

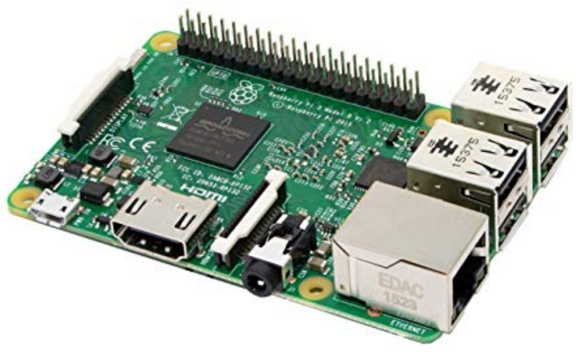


Fig.1 Raspberry Pi

Pi Camera: Pi Camera is compatible to raspberry pi for capturing photos and videos. It is pluggable and tiny device, which has 8 Mega Pixel sony image sensor installed in it. Camera is capable of recording 1080p video and can be managed through programming according to the requirements. Pi Camera is used as image detection sensor in the application. Whenever certain condition is met, pi camera

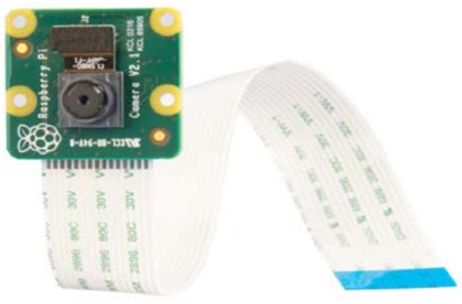


Fig.2 Pi Camera

triggers by the python script and captures the snapshot of the person.

Microphone: USB Microphone is also known as mic which is capable of listening audio from humans and transforms the sound to electrical signal. It is easy to plug into raspberry pi as it has USB connector. Microphone is used in the application because by default raspberry pi does not have any input microphone to receive audio input from users.

Fig.3 Microphone

Speaker: USB Speaker can provide audio output to outside world. It is compatible with raspberry pi and can convert text output to speech. Also, the reason for using speakers in the application is by default raspberry pi does not contain any speaker to provide audio output. In addition, adding speakers to the application makes audio output effective and loud to the user.

Fig.4 Speakers

Router: Router is the internet gateway which can be connected to raspberry pi for accessing internet through ethernet cable. Without internet router, the application may not function properly because some of the critical features like sending alerts to users through email dependent on internet router.

Fig.5 Router

Email: Email system can be set up in raspberry pi for sending notifications using Simple Mail Transfer Protocol (SMTP). Required library is configured in python code for accessing and managing the email notifications. Whenever a suspicious activity is detected by the application, it will send email notification to users.

Text Messaging: For managing the text message notifications, corresponding python libraries are used. Text

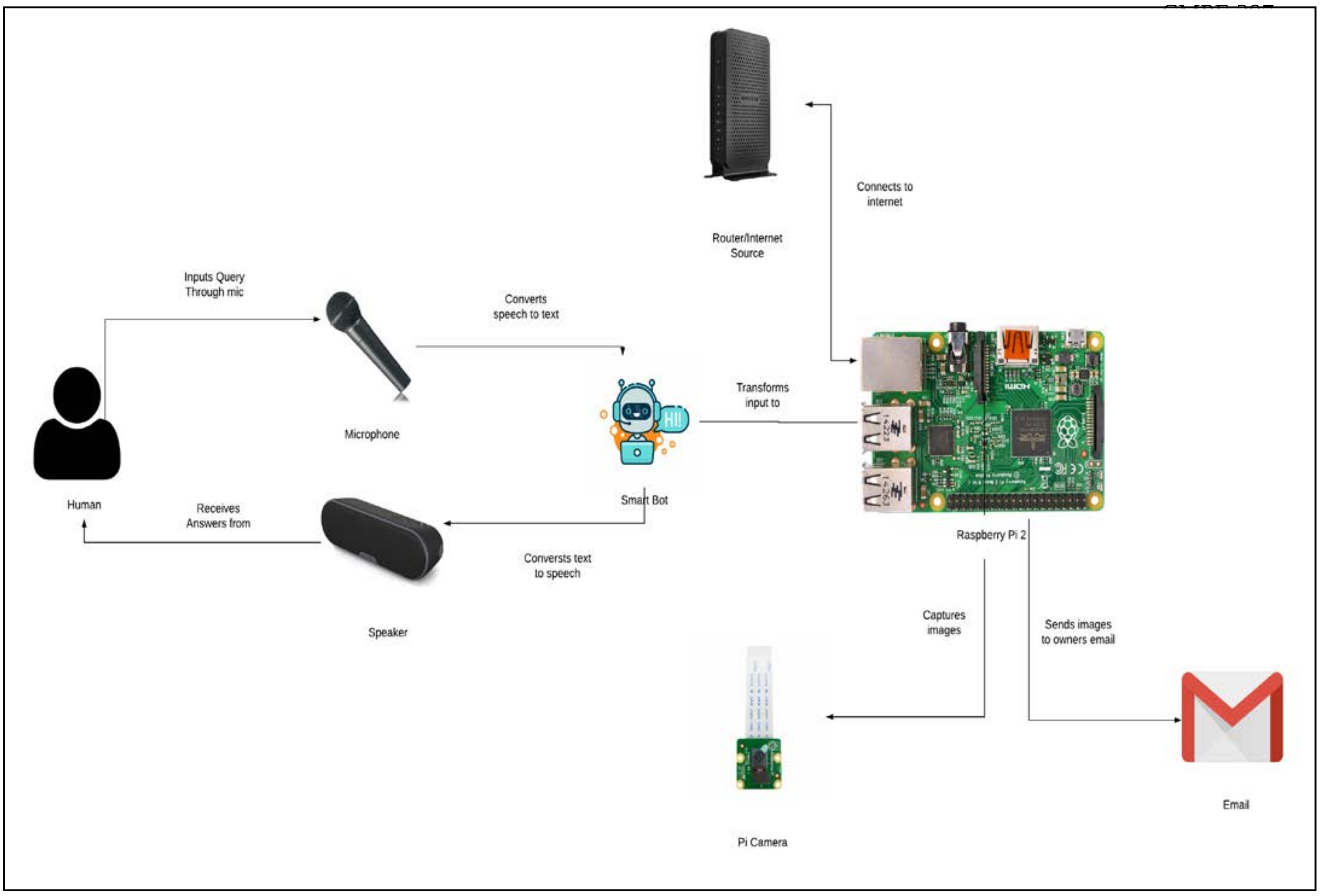


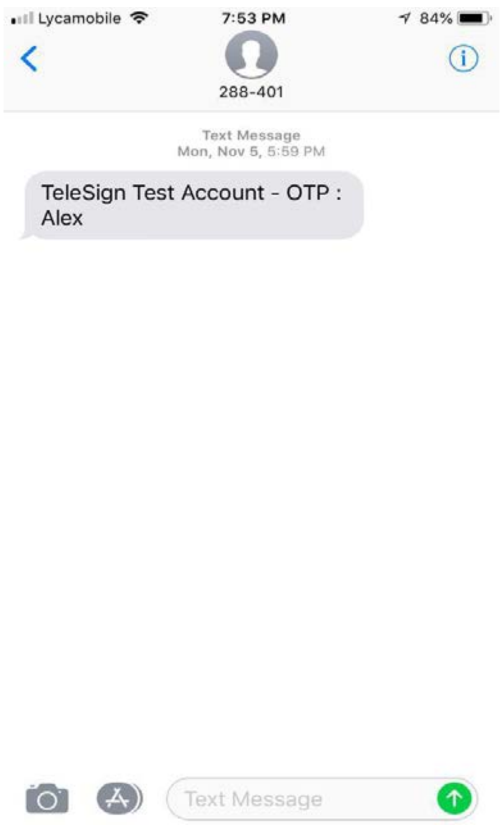
Fig.6 Architecture Diagram

messaging service is used to provide One Time Passwords to user.

B. Application Workflow (Desu, Sai Teja)

Following steps explains about the end to end system workflow when user interacts with application.

- Owner of the safe interacts with smart chatbot installed on raspberry pi through microphone using wake word (example: Hey Spartan).
- Microphone converts the user input into electrical signals and passes to the chatbot for parsing the human query.
- If audio of the human is not clear or understandable, smart chatbot asks human to input the query again.
- If audio is clear and understandable, chatbot triggers the actionable API to perform the requested action.
- Chatbot asks human “How can I help you?”
- Human then requests smart chatbot to “Unlock the safe”.
- Chatbot then parses the human input and triggers the required API, which generates the dynamic one-time password (OTP) and sends notification as email and text message to the owner.
- User or owner then provides the received OTP through voice to the application.


Lycamobile 7:53 PM 84%

288-401

Text Message
Mon, Nov 5, 5:59 PM

TeleSign Test Account - OTP : Alex

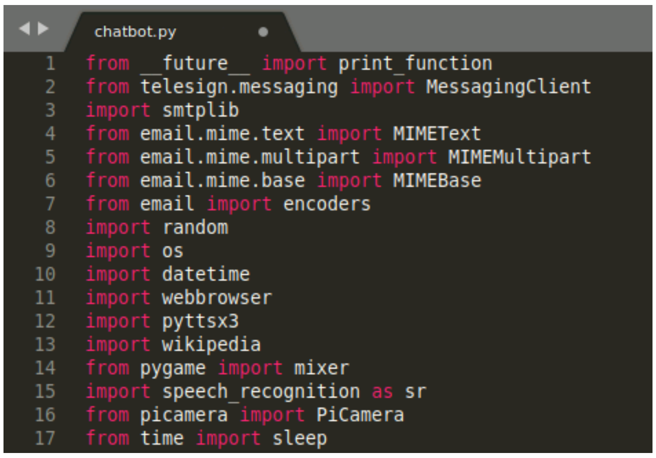
Fig.7 User Receiving OTP

- When unknown user tries to provide a random password, then chatbot detects the suspicious activity and triggers the camera module for capturing the photo of the person.
- Then, corresponding API, sends email to the owner notifying suspicious attempt unlocking the safe. Email consists of the captured photo when unknown person attempts to open the safe.

Fig.8 User Receives Intruder Alert Email

C. *Implementation (Bhavirisetty, Akhil)*

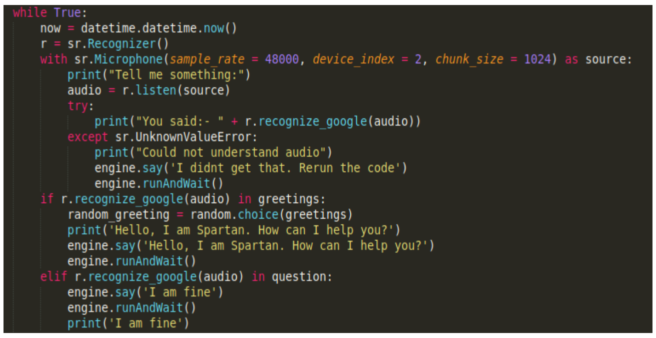
For developing the smart lock safe application, we have used majorly python scripting for training the model and responding to the user queries. In addition, configured multiple libraries for speech recognition, pi camera, email service (SMTP Protocol), and text messaging service. Following picture shows the number of libraries used for building the application.



```python
from __future__ import print_function
from telesign.messaging import MessagingClient
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders
import random
import os
import datetime
import webbrowser
import pyttsx3
import wikipedia
from pygame import mixer
import speech_recognition as sr
from picamera import PiCamera
from time import sleep
```

Fig.9 Python Libraries

For training the chatbot, python scripting is used. Application is trained extensively for possible user queries, to respond with appropriate answers. Besides training normal queries, application is also trained to behave normally if user asks any out of context queries. Below figures shows the sample loop to handle initial conversation with user.



```python
while True:
    now = datetime.datetime.now()
    r = sr.Recognizer()
    with sr.Microphone(sample_rate = 48000, device_index = 2, chunk_size = 1024) as source:
        print("Tell me something:")
        audio = r.listen(source)
        try:
            print("You said:- " + r.recognize_google(audio))
        except sr.UnknownValueError:
            print("Could not understand audio")
            engine.say('I didnt get that. Rerun the code')
            engine.runAndWait()
    if r.recognize_google(audio) in greetings:
        random_greeting = random.choice(greetings)
        print('Hello, I am Spartan. How can I help you?')
        engine.say('Hello, I am Spartan. How can I help you?')
        engine.runAndWait()
    elif r.recognize_google(audio) in question:
        engine.say('I am fine')
        engine.runAndWait()
        print('I am fine')
```

Fig.10 Sample Code

While testing the application, we can see the flow and response of the applications at the console of the python script. The following captured screenshot displays the happy flow of our application. The chatbot, greeted the user and provided OTP to user to provide as input to validate the identity. Application, then validates and unlocks the safe.

Fig.11 Voice Chat with Application

IV. System Testing (*Bhavirisetty, Akhil*)

Testing the smart safe lock is important to ensure every module is working fine. Following are the test strategies used for testing the application.

A. *Hardware Testing*

There are many hardware components in the application, which needs to be configured and integrated properly with raspberry pi.

a) Raspberry pi test: Ensured by installing the os multiple times and checked all the ports whether working properly or not.

b) Pi Camera test: Configured the camera module in raspberry pi to capture photos and record the video multiple times to ensure the same quality without any latency

c) Speaker test: Installed and configured the speaker device drivers and tested by playing pre-recorded audio multiple times to ensure the same audio output quality.

d) Microphone test: Ensured the audio input by recording the voice of human multiple times without any lag in the audio quality.

B. *Software Testing*

There are multiple software modules in this application which has been tested by unit testing and integration testing.

a) Unit Testing: Each individual module has been tested as per the test cases which consists expected outcome. In case of the failed scenarios, application module has been re-written to handle the corner cases.

b) Integration Testing: This testing is critical for our application because of multiple hardware components and software modules involved. After developing the application, tested from end to end which every module has been verified closely. In case of any communication loss between the modules, made sure the application system is reliable and robust to any interaction failures.

C. *Testing Tools (Singh, Vinitkumar Ramprakash)*

Below are the tools used for backend API's.

JMeter: JMeter is a tool which is used for testing the performance of each python API. JMeter is used specifically for unit testing for database connection with the backend server. In addition, used this tool for advanced monitoring statistics and it has lot of easy to use plugins for trying different types of testing.

Mocha: Mocha is a software tool which is used to test each functionality of python modules. Mocha is a great tool which has pretty rich set of features to test python framework. Mocha is very good in testing asynchronous calls of functions defined in the application.

Manual Voice Testing: As the application is voice chatbot, it has been tested with multiple varieties of humans who speak different dialect of English to identify if application is facing any issue. There are automated voice tools in the market to test the chatbot, nonetheless manual voice testing is more effective in achieving good results.

V. Results (*Singh, Vinitkumar Ramprakash*)

The result of the smart safe lock application is, owners can now able to unlock the safe by just providing OTP through voice and can also receive intruder alerts through email and text message. The important outcome of the application is enhanced user experience and consumes less time and efforts from user.

VI. Challenges (*Singh, Vinitkumar Ramprakash*)

While building this application, we faced following challenges. However, we have overcome the challenges by exploring different devices and configuring different software's.

- Defective raspberry pi
- Delicate set up such as inserting pi camera component
- Handling multiple dialects of English language
- Configuring up the email and message notification
- Performance of the chatbot in receiving questions and providing answers

VII. Future work (*Bhavirisetty, Akhil*)

The current application can handle the verification of owner and send notifications of unknown activity to user through multiple means of communication. However, this application has potential in making extraordinary product by including following features.

- Integrating the set up to the actual physical safe
- Authorize by face detection
- Authorize by voice detection
- Accessing the live streaming of the room when needed
- Making the set up more compact
- Improvising the aesthetics of the set up

VIII. Acknowledgment (*Singh, Vinitkumar Ramprakash*)

We would like to express our sincere thanks to Professor. Dr. Ammar Rayes for encouraging us to build a real time IoT project and followed many concepts in our application which are taught in class. Also, we would like to acknowledge the writers and authors of the reference articles which we have gained knowledge in implementing the application.

IX. Conclusion(Desu,Sai Teja)

At the end, smart safe lock is a secure, portable, and smart application which helps users in unlocking the safe with very less efforts. Also, at the same time unique features like voice chat, and notifications makes the application user friendly and likable.

References

[1] https://ieeexplore.ieee.org/document/8055694

[2] https://cloud.google.com/speech-to-text/docs/

[3] https://www.telesign.com/docs/

[4] https://standard.telesign.com/api-reference/apis/sms-api/overview

[5] https://github.com/TeleSign/python_telesign

[6] https://picamera.readthedocs.io/en/release-1.13/

[7] https://www.raspberrypi.org/documentation/