Smart Secured Wireless ATM Using Finger Print Recognition

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

In

Electronics and Communication Engineering

by

Sagar (16BEC0769)

Manjari Jain(16BEC0077)

Under the guidance of,

Dr. Saravanan K

SENSE

VIT, Vellore

April, 2020

DECLARATION

We hereby declare that the thesis entitled "Smart Secured Wireless ATM Using Finger Print Recognition" submitted by us, for the award of the degree of *Bachelor of Technology in Electronics and Communication Engineering* to VIT Vellore is a record of bonafide work carried out by us under the supervision of Dr. Saravanan K, SENSE, VIT, Vellore.

We further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date: Date Month 2020

Signature of the Candidates

Manjari Jain (16BEC0077)

Sagar (16BEC0769)

CERTIFICATE

This is to certify that the thesis entitled "Smart Secured Wireless ATM Using Finger Print Recognition" submitted by **MANJARI JAIN (16BEC0077) SENSE & SAGAR (16BEC0769) SENSE**, VIT, Vellore for the award of the degree of *Bachelor of Technology in Electronics and Communication Engineering*, is a record of bonafide work carried out by them under my supervision during the period, 01.12.2019 to 30.04.2020, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in our opinions, meets the necessary standards for submission.

Place: Vellore

Date:

Signature of the Guide

Internal Examiner **External Examiner**

Head of the Department Communication Engineering

Dr. Thanikaiselvan V,

Bachelor of Technology in Electronics and Communication Engineering

ACKNOWLEDGEMENT

With immense pleasure and a deep sense of gratitude, we wish to express our sincere thanks to our supervisor **Prof. Saravanan K.**, SENSE, VIT, without his motivation and continuous encouragement, this research would not have been successfully completed.

We are grateful to the Chancellor of VIT Vellore, **Dr. G. Viswanathan**, the Vice Presidents, for motivating us to carry out research in the VIT Vellore and also for providing us with infrastructural facilities and many other resources needed for my research.

We express our sincere thanks to **Dr. Kittur Harish Mallikarjun**, Dean, SENSE, VIT Vellore, and **Dr. Thanikaiselvan V**, HOD, SENSE, VIT for their kind words of support and encouragement. We like to acknowledge the support rendered by **our friends and faculties** in several ways throughout our research work.

We wish to extend my profound sense of gratitude to **our parents** for all the sacrifices they made during our research and also providing us with moral support and encouragement whenever required.

Place: Vellore

Date: 29th April 2020

MANJARI JAIN (16BEC0077)

SAGAR (16BEC0769)

Executive Summary

ATMs have become an essential part of the individual's daily routine as it is utilized to check one's amount balance and its most important function is to extract one's money. Nowadays, the culprits have the latest technologies for easily hacking into the secured systems of the banks and collect the confidential information of the clients, To counter that, fingerprint sensing incorporated with OTPs has been suggested, as it is globally accepted that the fingerprints of every person are unique and different, while OTPs don't hold its value like ATM PINs. This research is based on using Python GUI as the ATM screen. The innovation in this study exists in two ways. The first one is that OTPs will be sent via Python GUI, on the client's registered email address also, so that OTPs can still be accessed in case of SIMs lost. The second one is that including a URL: www.msbank.co.in for online enrollments of the clients and producing APIs. The main idea is to first check the client's fingerprints and then to verify the OTPs from our Mongo Database. The involved algorithm also maintains a check that the same email address cannot be utilized again for registration.

CONTENTS

List of Figures

List of Tables

Lists of Abbreviations

DB	Database
OTP	One Time Password
NO	Number
SQL	Structure Query Language
NOSQL	Not only Structure Query Language
ID	Identity
PIN	Personal Identification Number
ATM	Automated Teller Machine
SIM	Subscriber Identity Module
GSM	Global System for Mobile
2G	Second Generation
3G	Third Generation
4G	Fourth Generation
5G	Fifth Generation
API	Application Program Interface
URL	Uniform Resource Allocator

1 Introduction

In today’s world, people are using computers to store huge data sets. Data is inexpensively transmitted through the whole wide world. We are using ATM for approximately more than 50 years since its inauguration. From its launch to its present date, ATM has created an environment of withdrawing money without any difficulties.

In the area of banking, the flexibility of ATM to be deployed in every location possible is considered to be of eminence because that gives the liberty to the clients to make transactions at their end. The card reader and keys act as input gadgets while the visual screen and cash dispenser act as output tools. The bank-owned host processor acts as a medium channel for numerous ATMs to collaborate with them on the common level. [2]. Authorized Banks generate distinguished ATM PINs for their clients. These ATM PINs are sent through a registered phone number. If someone knows the user’s card details and saw the notification of PIN on a user’s phone. Then, they have the full accessibility to the client’s account and can be responsible for any illegalities caused.

Criminals have contrived different strategies to take cash from ATMs. Since they can’t simply remove cash from an ATM because of intensely made sure about the money compartment, they rather take card data of different clients. Taking ATM card data is a lot simpler than breaking into the ATM itself. Card skimming is one of these techniques hoodlums regularly use. Card skimmers are introduced on ATMs card per user space. They have an electronic circuit to peruse the data on an attractive piece of the card and this data is put away on card skimmer’s memory. This data is later replicated by the lawbreakers in the wake of expelling the skimmer from the ATM. Card skimmers are intended to be an ideal fit over unique cards per user so clients can without much of a stretch fall prey to it. On the off chance that card spaces feel massive or free, there are chances that it is a skimmer gadget.

There are likewise phony key cushions accessible in obscurity web commercial centers over the web, which can be put over the first key cushion of the ATM, which logs all the client inputs. These logged inputs are then abused by crooks to take cash from the client account. Fraudsters can likewise introduce a little camera simply over the keypad of the ATM, which continues recording client inputs, that incorporate your ATM PIN too. Clients are regularly exhorted by their banks to shroud keypad with their other hand while entering the PIN. At times, the whole front of the ATM is supplanted with a phony front, which looks precisely like a genuine one. If there should be an occurrence of a phony ATM front, it turns out to be difficult to know whether the ATM has been undermined. Cash doesn’t apportion out of the phony front and it likewise continues recording client inputs.

1.1 Objective

Our objective is to acquaint a powerful method to improve the existing situation of ATM by incorporating fingerprint sensing. It is globally accepted that the fingerprints of every individual are unique and different from one another. So, in this way, we will reduce the chances of thefts happening these days, as the culprits have the technologies to hack into the systems and gleaned all the confidential information of the user but he will not able to complete the transaction without the authentication of the fingerprint. We are also including One Time Password send via registered user's email address and phone number as ATM PIN to perform activities only after thorough **authentication of the user's fingerprint** from our database.

Fingerprint Authentication Overview

- Stage 1: Enrollment => The enrollment works in a way that first it encounters a fingerprint from the user, processes it to a digital image, purged the undesired features from it. Post-process the image and stored it in the database.
- Stage 2: Identification => We used a fingerprint sensor for operating on a fingerprint, it forms the feature extraction from it, matched it with the N templates of a fingerprint from our database.
- Stage 3: Verification => On matching of the fingerprint with a particular fingerprint from our database, we verified it again by running the same fingerprint with the matched database again.

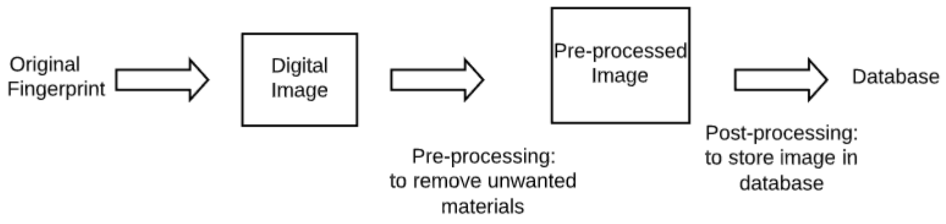

Original Fingerprint
Digital Image
Pre-processed Image
Database
Pre-processing: to remove unwanted materials
Post-processing: to store image in database

Fig 1.1. Enrollment [1]

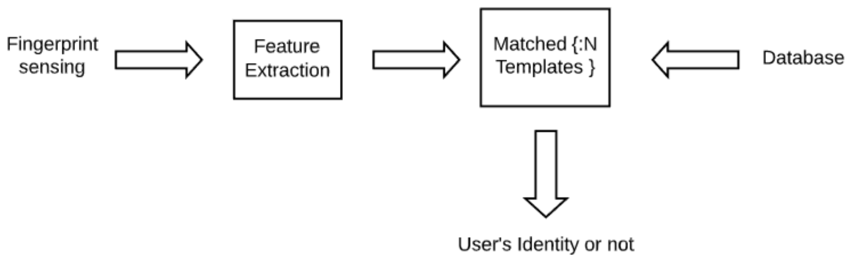


Fig 1.2. Identification [1]

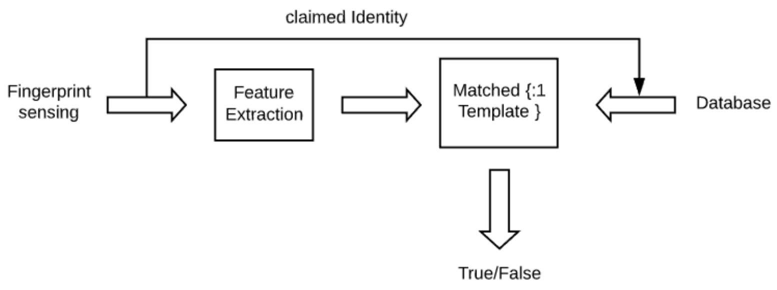


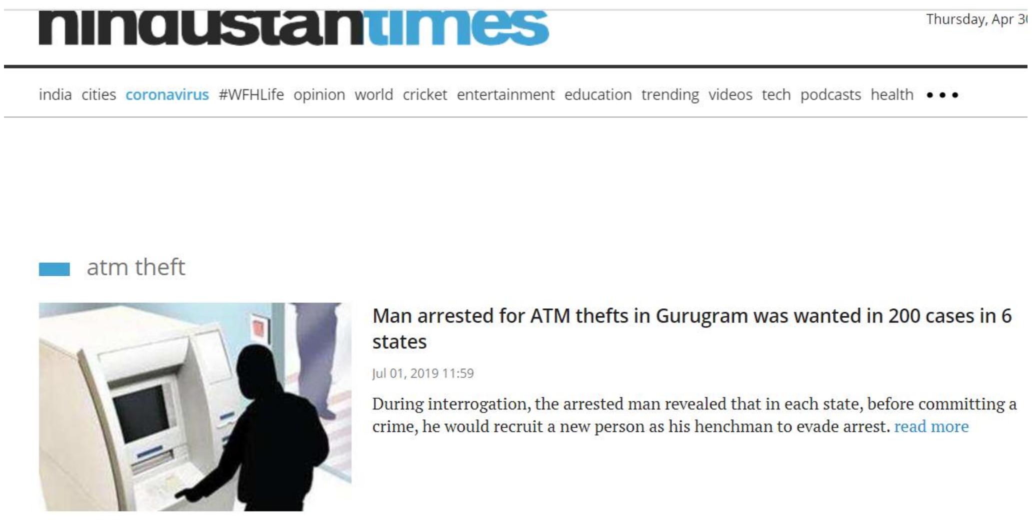
Fig 1.3. Verification [1]

Our Project Demonstration Overview:

On of top of the fingerprint recognition, three One Time Passwords will be sent to the matched fingerprint customer's registered Email ID customeruser@example.com along with the registered Phone Number 99XXXXXXXX. If he can enter the OTPs correctly on or before the third OTP is transferred, then, by all means, he is eligible to perform transactions. If he enters the amount to be withdrawn higher than the amount present in his bank account, then he has to start the process from the beginning. On completion of the transaction, he will receive a text message on phone as well as his email stating that the transaction was successfully on this Account No: 56XXXXXX with the remaining balance. On the other side, after the failure of the third OTP, he will receive an alert security breach email and will be asked to contact his nearest branch as soon as possible.

1.2 Motivation

Our motivation behind this research is that we came across many ATM theft reports regarding cloning cards, self-guessing PIN, etc.,

1. 
hindustantimes
Thursday, Apr 3
india cities coronavirus #WFHLife opinion world cricket entertainment education trending videos tech podcasts health •••
atm theft

Man arrested for ATM thefts in Gurugram was wanted in 200 cases in 6 states

Jul 01, 2019 11:59

During interrogation, the arrested man revealed that in each state, before committing a crime, he would recruit a new person as his henchman to evade arrest. read more

Fig 1.4. ATM Theft Report 1

Here is the detailed report of it from Hindustan Times:

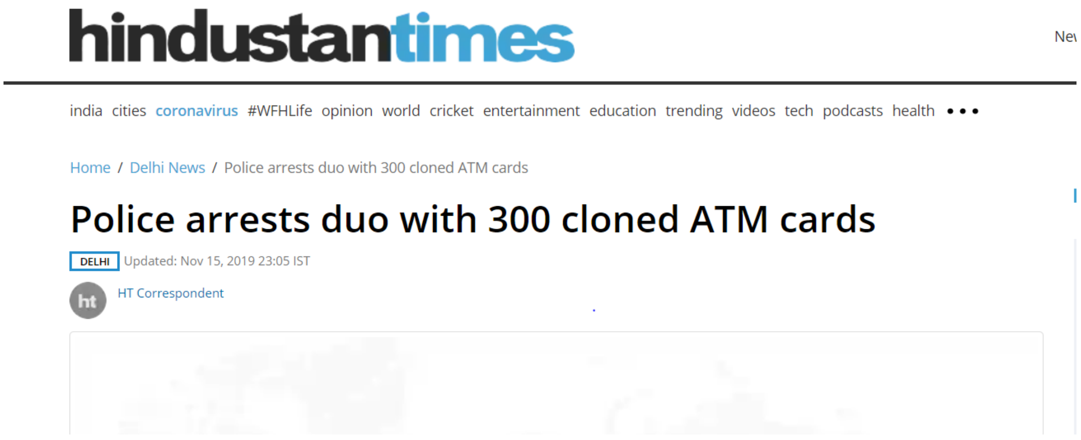
“Five days after the police captured a hoodlum needed in more than 200 instances of vehicle grabbing, ATM robberies, murder, endeavor to kill, plunder, grabbing, and attack, the police on Sunday said the charged had uncovered how he got away from the police for such a long time.

During cross-examination, he uncovered that in each state, before carrying out a wrongdoing, he would enlist a renewed individual as his associate to sidestep capture.

Charged Shahid false name Advani, pioneer of infamous Advani group, was supposedly needed for contribution in criminal cases in Haryana, Delhi, Rajasthan, Gujarat, UP and Maharashtra, the police said. He worked in evacuating ATMs and cut open the money plate with a gas shaper, the police said.

City police representative Subhash Boken said for every one of his exercises Advani would lease vehicles at above-advertise rates to win the proprietor’s trust. “He used to do a recce of the stand to be plundered. It would either be in a disengaged place or unmanned,” he said.”

2.


hindustantimes

india cities coronavirus #WFHLife opinion world cricket entertainment education trending videos tech podcasts health

Home / Delhi News / Police arrests duo with 300 cloned ATM cards

Police arrests duo with 300 cloned ATM cards

DELHI Updated: Nov 15, 2019 23:05 IST

HT Correspondent

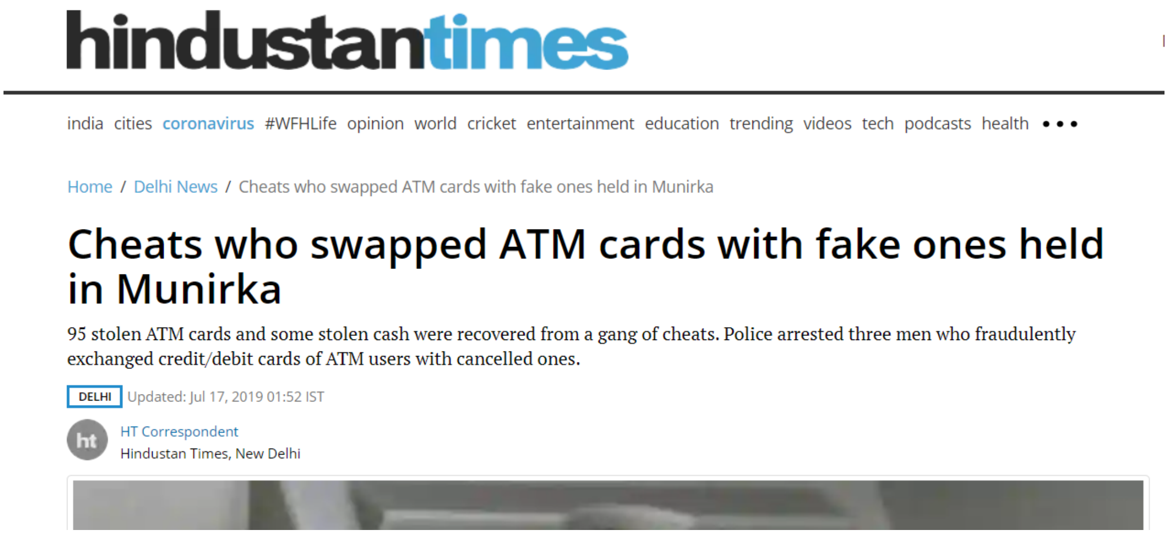
Fig 1.5. ATM Theft Report 2

Here is the detailed report of it from Hindustan Times:

“Having bought a skimming gadget and a government agent camera from a Chinese web-based business site, two claimed fraudsters focused on ATMs in the national capital and seven different states to hoodwink many individuals, the Delhi Police said on Friday in the wake of capturing them and recouping more than 300 cloned cards from their ownership.

While the pair, Krishna Gopal and Sumit Gola, had figured out how to fix the skimming gadget and camera to ATMs, they didn't have the foggiest idea how to decode the taken information and clone cards. "Contingent upon how much cash they could make from each card, the fraudsters were paying programming engineers anyplace somewhere in the range of ₹100 and ₹12,000 for cloning each card," a senior cop said.

Delegate official of police (south-west area), Devender Arya, said the police got a tip that helped capture the two suspects from south Delhi's Mahipalpur. " We requested that they stop when we caught them yet they began to run. We had the option to pursue and catch them. Their pack contained more than 300 cloned charge and Mastercards, a skimming gadget, a camera, and different things utilized in the misrepresentation," DCP Arya said."

3. 

hindustantimes

india cities coronavirus #WFHLife opinion world cricket entertainment education trending videos tech podcasts health •••

Home / Delhi News / Cheats who swapped ATM cards with fake ones held in Munirka

Cheats who swapped ATM cards with fake ones held in Munirka

95 stolen ATM cards and some stolen cash were recovered from a gang of cheats. Police arrested three men who fraudulently exchanged credit/debit cards of ATM users with cancelled ones.

DELHI Updated: Jul 17, 2019 01:52 IST

HT Correspondent
Hindustan Times, New Delhi

Fig 1.6. ATM Theft Report 3

Here is the detailed report of it from Hindustan Times:
"Delhi police have busted a pack of cheats and captured three men who deceitfully traded credit/check cards or ATM clients with dropped ones and later pull back cash utilizing them.

Upwards of 95 taken ATM cards and some taken money were recuperated from them. The three men were captured following a pursuit in south Delhi's Munirka on Monday night, during which one of them started shooting, police said.

Police distinguished the posse chief boss as Mohammad Sakib, a BSc graduate associated with at any rate five instances of cheating and robberies in Telangana. His two partners are Adil from Ballabhgarh and Abid, a grappler from Palwal.

Delegate officials of police (wrongdoing) Rajesh Deo said various individuals as of late announced they were focused by a group that duped them into changing their ATM cards and pulled back cash later.”

In the majority of the reports, we found that the culprits can clone the cards with the help of Software Engineer. They were able to decode the ATM PIN by using various permutations and combinations of numbers and ultimately hitting their correct guess. That’s why we studied various research papers regarding how to make ATM more secure. Fingerprint authentication is impeccable to be hacked by the computer systems so it makes its value to be of high eminence. Thus, incorporating it with ATM along with random passwords generated, it would be impossible to get a hold of it.

Various methods have come to neutralize this current situation, one of them is face recognition fails because due to aging in the human biological body, the various facial features grow out of proportion of the last image taken of the user. The user has to re-registered his/her image periodically to sustain its productivity, which could be a tremendous effort for a bed-ridden patient or face-burnt victims.

Using biometric identifiers offers many benefits over traditional and modern methods. It is a very undemanding job to install as it takes a little time and effort to find the individual fingers and fingerprint scanner. Therefore, fingerprint recognition is considered among the most accessible in all biometric verification techniques.

1.3 Background

Korea welcomed its first ATM in 1975. With the help provided by the Shinhan Bank, the locals were able to utilize the advantages of an ATM. [5].

According to [6], there was a webcam installed on the ATM which stores the live photo-age of any clients that walk by. This captured image is then contrasted with the remaining images from the database. The successful result of the contrast generates an arbitrary password that is delivered on the client’s registered phone. Then the client is instructed to enter the same password on the ATM screen. If the user enters accurately then he/she is able to perform transaction operations. But this theory has two major flaws: 1) If the mobile phone is lost or stolen then it will become difficult to prevent theft from happening until unless immediate action is taken. 2) With aging, our facial expressions tend to change, hence the original image captured at that time will be compared with the latest image of our faces which will lead to discrepancies.

The Palm Print Recognition and Confirmation System used by the ATM [7], the idea of using the PIN as a secure password is not highly suitable by the researchers. Therefore, in order to broaden the security, incepting the concept of palmprint authorization was introduced. The profound research was that a simulator imitates the current ATM functions. In concluding, they compared 89.43% of the cervical recognition program with a rejection rate of 10.57% [7]. Therefore, 53 out of 500 customers who visit the upgraded ATMs through this verification process might have difficulties. The rejection rate was elevated as high as ten percent.

The Graphical User Authentication introduced by Lalzirtira was to purge mistakes in the alphanumeric certification of ATMs. The work done by him suggested that a code that incorporates image is more straightforward. He came up with a password that would aid clients to memorize it easily in their minds [8]. The pass-code which includes images for providing authorization in the ATM system has its own advantages in some levels but failed due to video recordings.

Another theory that was established was of dyno-passwords. Here the client is provided with a text message from the bank on their registered phone number containing the passcode. This new passcode will be entered through the ATM screen.

The database server of the bank will again cross-examine it. Thus, implying to withdraw money from ATM, all one needs is the client's PIN, card, and SIM. Anyone in the social circle of the client can be easily equipped with this amount of confidential information. As a result of this, this theory opened the doors for hackers [9].

According to [10], other neural network-based research work has been pursued to match clients' fingerprints through the method of its patterns of ridges established. This theory was achievable only on binary images; On the other hand, this research states that once a group is traced, then it can be traced with utmost efficiency. This theory fails on the grounds of unreachable neural networks [10].

2 Project Description and Goals

The present ATM works in such a way that the person first enters the ATM. He/she inserts his/her ATM card into the machine. Then, they enter the ATM PIN for verifying that they are the owner of the ATM Card. After the verification, they select the type of Accounts from which they want to withdraw money. If they have sufficient balance in their account, the ATM machine dispenses the amount and informed them via a Text Message / Email about the transaction successful and remaining balance present in the amount.

A computerized technology that establishes the role of being a monetary medium between the clients and the bank to maintain the functionality of transactions without the presence of a third party. Some charge cards, be that as it may, may experience more difficulty. ATMs are highly feasible in nature because it allows the clients to perform any required transaction without having the obligation to go to the bank.

The PIN stands for Personal Identification Number. It is a combination of four numbers, each one ranging from 0 to 9 amounts to a decade of PINs possible, which is preferably set by the clients. This PIN remains universal throughout the longevity of the bank account until unless it is not changed by the client. Since it is modicum in size, it can be swiftly memorized. The hacker has every technological tool at their disposal to guess the accurate PIN [12][13]. When the client enters the PIN on the ATM screen then it is cross-examined from the database of the bank organizations.

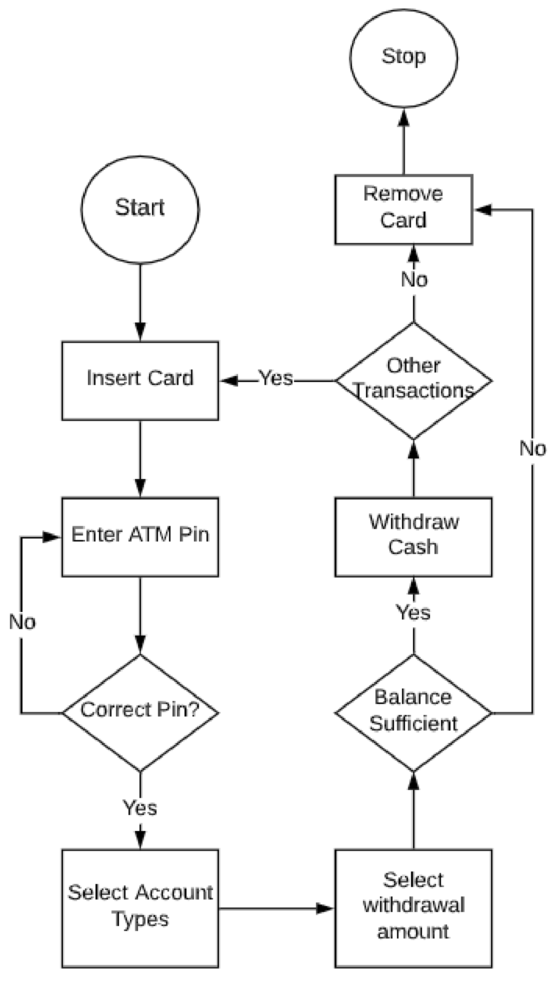


Fig 2.1 Present ATM Workflow

Conventional validation frameworks (utilization of PIN) can't separate between an impostor who falsely gets the entrance benefits (card and PIN) and the veritable client [26]. In this manner, to pick up the ATM machine client's certainty, a second level biometric validation security must be set up related to the previously existing individual recognizable proof number (PIN).

Our description is to use fingerprint authentication as one of the major keys to our research work.

Biometrics is an innovation that assists with making our information immensely secure, separating all the clients by method for their own physical attributes. Biometric data can be utilized to precisely recognize individuals by utilizing their unique finger impression, voice, face, iris, penmanship, or hand geometry, etc. Utilizing biometric identifiers offers a few points of interest over conventional and current strategies. Tokens such as attractive stripe cards, shrewd cards, and physical keys, can be taken, lost, copied, or left behind; passwords can be mutual, overlooked, hacked or accidentally saw by an outsider [27].

There are two key capacities offered by a biometric framework. One technique is ID and the other is confirmation. In this research, we are focusing on distinguishing and confirming a client by unique mark acknowledgment. An advanced ATM is commonly comprised of the gadgets like CPU to control the UI what's more, gadgets identified with an exchange, Magnetic or Chip card per user to distinguish the client, PIN Pad, Secure crypto-processor for the most part inside a protected spread, Display to be utilized by the client for playing out the exchange, Function key catches, Record Printer to furnish the client with a record of their exchange, to store the pieces of the apparatus requiring confined access – Vault, Housing for feel, Sensors and Indicators [28]. Unique finger impression innovation is the most generally acknowledged and develop biometric strategy and is the most effortless to convey and for a more significant level of security readily available. It is easy to introduce and furthermore it requires some investment and exertion to gain one's unique finger impression with a unique finger impression recognizable proof gadget. In this manner, unique finger impression acknowledgment is considered among the least meddlesome of all biometric confirmation procedures. Old occasions authorities utilized thumbprints to seal archives a large number of quite a while back and law offices have been utilizing unique mark recognizable proof since the late 1800s [36].

Our project will use the Mongo DB for the storage of confidential information about the customers. Fingerprint Sensor R3 07 will be attached to the Arduino board that will be used for authenticating the fingerprints of each individual that enters the ATM Machine. Customers will enter their unique information on the ATM Screen in the form of the Python GUI. After careful registration, they will be asked to enter fingerprints again. When their fingerprints are successfully authorized then a small line will appear on the Python GUI stating that "Successfully fingerprint verified. Now you can withdraw the money". A new popup for entering the OTP will appear in front of them. Whenever they enter the wrong OTP, another small line will appear on the Python GUI stating that "Wrong otp ${otp_number}". After entering inaccurately for the third time, then an email will be sent to their registered email address with the Subject of "Security Theft Alert, kindly contact your nearest branch as soon as possible". In any of the three times, the customer enters the OTP correctly, then they will be proceeding to a new box stating the "Enter the amount". If they enter the amount which is insufficient from the balance present in their account then they will be asked to restart the process all over again. On the other hand, if they enter the amount which suffices in terms of the balance present in their accounts, then the money will be withdrawn and our Database will automatically update the remaining balance. It will also send out the mail with the subject line of "Successful Transaction on this Account No.99XXXXXXX".

Our goal is to reduce the number of thefts occurring due to the loss of ATM cards, embezzling classified data of clients regarding their secured ATM PINs. Thus, by using fingerprint authorization, the chances of these thefts occurring will be shrunk in a significant amount. As the hackers don't have the technologies that could sustain the fingerprints in the uttermost HD quality. Secondly, the ATM PINs which become universal to all the activities linked to the transaction of anything through the bank would not be so consistent. As we are using One Time Passwords to demise the comprehensive consequences of having the standard ATM PIN to be used everywhere.

3 Technical Specification

(a) Hardware:

- <u>Arduino Uno</u>

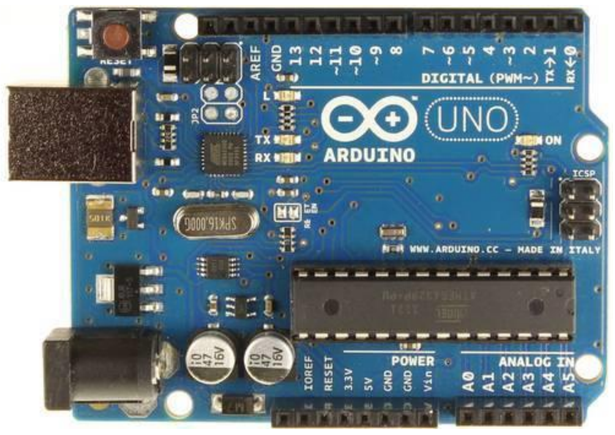


Fig 3.1 Arduino Uno

The Arduino Uno is a microcontroller board dependent on the Atmega328. It has 20 computerized input/yield pins (of which 6 can be utilized as PWM yields and 6 can be utilized as simple data sources), a 16 MHz resonator, a USB association, a force jack, an in-circuit framework programming (ICSP) header, and a reset button. It contains everything expected to help the microcontroller; basically, associate it to a PC with a USB link or force it with an AC-to-DC connector or battery to begin.

The Uno varies from every single going before the board in that it doesn't utilize the FTDI USB-to-sequential driver chip. Rather, it includes an Atmega16U2 modified as a USB-to-sequential converter. This assistant microcontroller has its own USB bootloader, which permits propelled clients to reconstruct it.

The Arduino has an enormous help network and a broad arrangement of help libraries and equipment add-on "shields" making it an incredible basic stage for inserted hardware.

This is the third update of the Uno, which has various changes:

- The USB controller chip changed from Atmega8U2 (8K streak) to Atmega16U2 (16K blaze). This doesn't expand the blaze or RAM accessible to portrays.
- Three new pins were included, which are all copies of past pins. The I2C pins (A4, A5) have been additionally been brought out on the board close AREF. There is an IOREF pin close to the reset pin, which is a copy of the 5V pin.
- The reset button is presently close to the USB connector, making it increasingly open when a shield is utilized.

➢ <u>Fingerprint Sensor R3 07:</u>

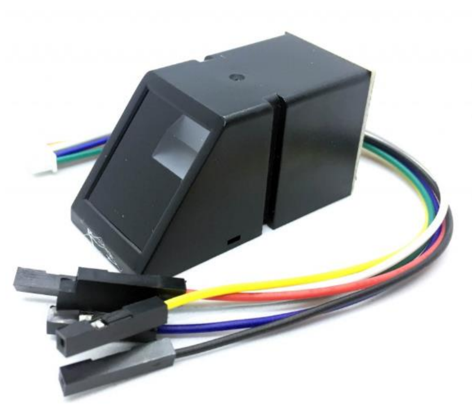


Fig 3.2 Fingerprint Sensor R3 07

The R307 unique mark module has two interface TTL UART and USB2.0, USB2.0 interface can be associated with the PC; RS232 interface is a TTL level, the default baud rate is 57600, can be changed, allude to a correspondence convention; can And microcontroller, for example, ARM, DSP and other sequential gadgets with an association, 3.3V 5V microcontroller can be associated legitimately. Necessities to interface the PC level change, level transformation note, encapsulations, for example, a MAX232 circuit.

Highlights:

- Against static capacity: a solid enemy of static capacity, hostile to static file arrived at 15KV above.
- Application improvement is basic: engineers can give control guidelines, self-unique mark application item advancement, without the requirement for proficient information on fingerprinting.
- Customizable security level: appropriate for various applications, security levels can be set by the client to alter.
- Finger contact detecting signal yield, low powerful, detecting circuit backup current is low, under 5uA.
- Immaculate capacity: autonomous unique mark assortment, finger impression enlistment, unique mark examination (1: 1), and unique mark search (1: N) work.
- Little size: little size, no outer DSP chip calculation, has been incorporated, simple to introduce, less shortcoming.
- Ultra-low force utilization: low force utilization of the item in general, appropriate for low-power prerequisites of the event.

Technical Parameters:

- Supply voltage: DC 4.2 ~ 6.0V
- Supply current: Working current: 50mA (typical) Peak current: 80mA
- Fingerprint image input time: <0.3 seconds
- Window area: 14x18 mm
- Matching method: Comparison method (1: 1)
- Search method (1: N)
- Characteristic file: 256 bytes
- Template file: 512 bytes
- Storage capacity: 1000 pieces
- Security Level: Five (from low to high: 1,2,3,4,5)

- Fake rate (FAR): <0.001%
- Refusal rate (FRR): <1.0%
- Search time: <1.0 seconds (1: 1000 hours, mean value)
- Host interface: UART \ USB1.1
- Communication baud rate (UART): (9600xN) bps Where N = 1 ~ 12 (default N = 6, i.e. 57600bps)
- Working environment: Temperature: -20 ℃ - +40 ℃ Relative humidity: 40% RH-85% RH (no condensation)
- Storage environment: Temperature: -40 ℃ - +85 ℃ Relative humidity: <85% H (no condensation)

(b) Software:

- Mongo DB:

Fig 3.3 Mongo DB

MongoDB is an open-source record database that gives superior, high accessibility, and programmed scaling. In straightforward words, you can say that – Mongo DB is a record situated database. It is an open-source item, created and bolstered by an organization named 10gen. MongoDB is accessible under General Public permit for nothing, and it is additionally accessible under Commercial permit from the producer.

“Mongo DB can be defined as a document-oriented database system that uses the concept of NoSQL. It also provides high availability, high performance, along with automatic scaling. This open-source product was developed by the company – 10gen in October 2007, and the company also maintains it. MongoDB exists under the General Public License (GPL) as a free database management tool as well as available under Commercial license as of the manufacturer. MongoDB was also intended to function with commodity servers.

Companies of different sizes all over the world across all industries are using MongoDB as their database."

In MongoDB, a database can be characterized as a physical compartment for assortments of information. Here, on the record framework, each database has its assortment of documents living. Normally, a MongoDB server contains various databases.

➢ Python Spyder:

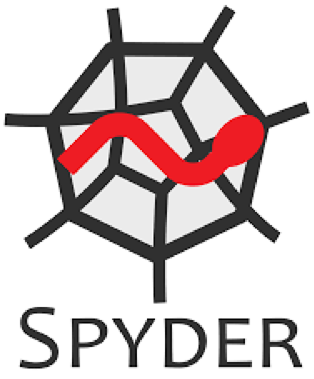


Fig 3.4 Python Spyder

Python is a clear and powerful object-oriented programming language, comparable to Perl, Ruby, Scheme, or Java.

Some of Python's notable features:

- Uses an elegant syntax, making the programs you write easier to read.
- It is an easy-to-use language that makes it simple to get your program working. This makes Python ideal for prototype development and other ad-hoc programming tasks, without compromising maintainability.
- Comes with a large standard library that supports many common programming tasks such as connecting to web servers, searching text with regular expressions, reading, and modifying files.
- Python's interactive mode makes it easy to test short snippets of code. There's also a bundled development environment called IDLE.
- Is easily extended by adding new modules implemented in a compiled language such as C or C++.
- Can also be embedded into an application to provide a programmable interface.

- Runs anywhere, including Mac OS X, Windows, Linux, and Unix, with unofficial builds also available for Android and iOS.
- Is free software in two senses. It doesn't cost anything to download or use Python, or to include it in your application. Python can also be freely modified and re-distributed because while the language is copyrighted it's available under an open-source license.

➢ <u>Tkinter:</u>

Fig 3.5 Tkinter

Python offers multiple options for developing a GUI (Graphical User Interface). Out of all the GUI methods, Tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with Tkinter is the fastest and easiest way to create GUI applications. Creating GUI using Tkinter is an easy task.

To create a Tkinter app:

- Importing the module – Tkinter
- Create the main window (container)
- Add any number of widgets to the main window
- Apply the event Trigger on the widgets.

➢ <u>Visual Studio (Node JS):</u>

Fig 3.6 Visual Code

Node.js is a platform for building fast and scalable server applications using JavaScript. Node.js is the runtime and npm is the Package Manager for Node.js modules.

Visual Studio Code has support for the JavaScript and TypeScript languages out-of-the-box as well as Node.js debugging. However, to run a Node.js application, you will need to install the Node.js runtime on your machine.

To get started in this walkthrough, install Node.js for your platform. The Node Package Manager is included in the Node.js distribution. You'll need to open a new terminal (command prompt) for the node and npm command-line tools to be on your PATH.

4 Material, Methods and Details

Our initiated methodology is to include fingerprint recognition along with the One Time Passwords. Hence, we are appending one more layer of secured and personal identification in the present system.

4.1 Materials and Methods

Materials used:

- Fingerprint Sensor Adafruit R307
- Arduino Uno
- Power Supply
- Connecting Wires
- Python
- Spyder
- Tkinter
- Mongo DB
- Visual Code (Node JS)

The method behind fingerprint recognition:

The most widely accepted security measure to be installed in the ATM is of biometric fingerprint technology. It delivers a much more secure initiative than the existing ATM security. The quality of this technology to be elegantly aligned with our present lives is quite impressive as it is quite undemanding to install and to be operated at. Due to this, it is accoladed as one of the most successful authentication techniques. The fingerprints data are not stored in any database; thus, no misuse of the fingerprint is conceivable.[24]

The unique finger impression ID depends on two fundamental suppositions: - Invariance and Peculiarity Invariance. Invariance implies the finger impression qualities don’t change along with life. Peculiarity: implies the unique mark is extraordinary and no two people have a similar example of a unique finger impression.

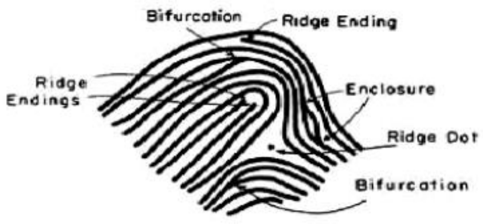


Fig 4.1 Features of a fingerprint [1]

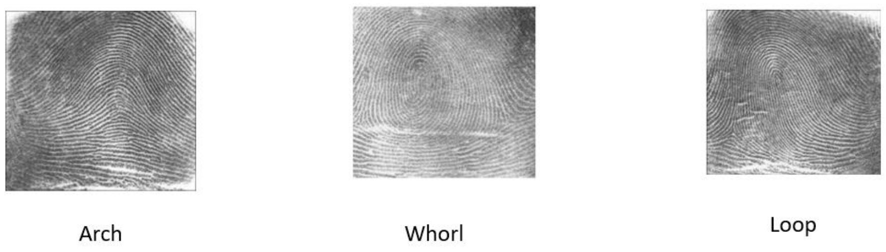


Fig 4.2 Types of fingerprints

Process

- Unique biometric impression: The unique pattern is the component example of the biometric. [2]
- Binarization: It describes the changes from greyscale into binary picture by finalizing the esteem value. [2]
- Block Filter: It is the method of diminishing the thickness of all ridgelines to a solitary pixel width to extricate minutiae viably. [2]
- Minutiae Extractruction: Details are inferred following minutiae removal.[2]
- Minutiae Matching: It is to compare the data of distinctive finger impressions. The layout of minutiae coordination is also employed. [2]

- Matching Score: It is the relation to calculate the collaborating score of the original fingerprint & template data. [2]

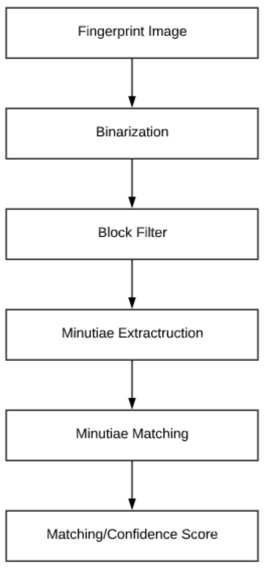


Fig 4.3 Fingerprint Authentication Process Diagram [2]

The framework of our method to improvise the condition of ATM

Only after careful verification of the clients' fingerprints, they will proceed to their registered Email IDs as well as registered Phone Number to check for One Time Password from the bank. They will be given three chances to write their correct OTP in front of the ATM screen. If they enter it accurately, then they will be permitted to withdraw money on the contingency of having sufficient balance in their accounts. On the other hand, even after providing them with three One Time Passwords, they enter incorrectly then they will receive a mail from us stating of "Security Breach and Kindly Contact your nearest branch". We will be using APIs to send and receive data of the uniquely identified customer.

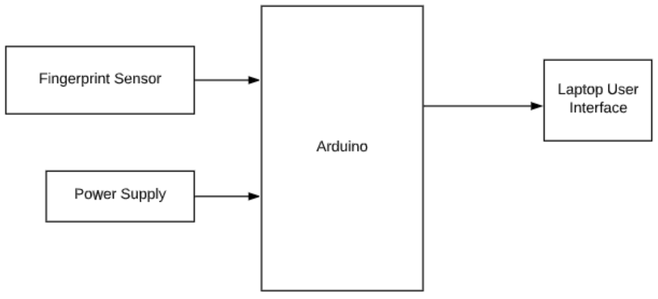
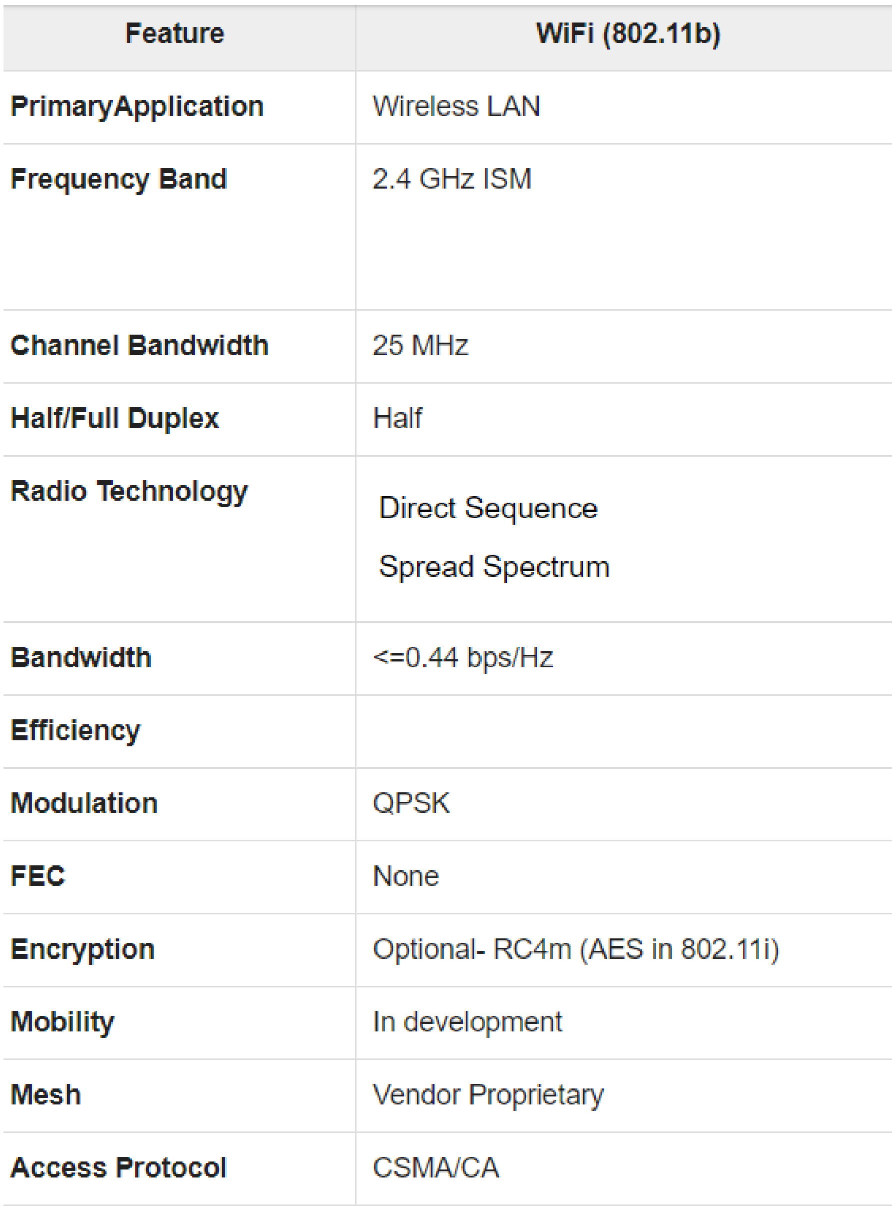


Fig 4.4 Block Diagram

4.2 Code and Standards

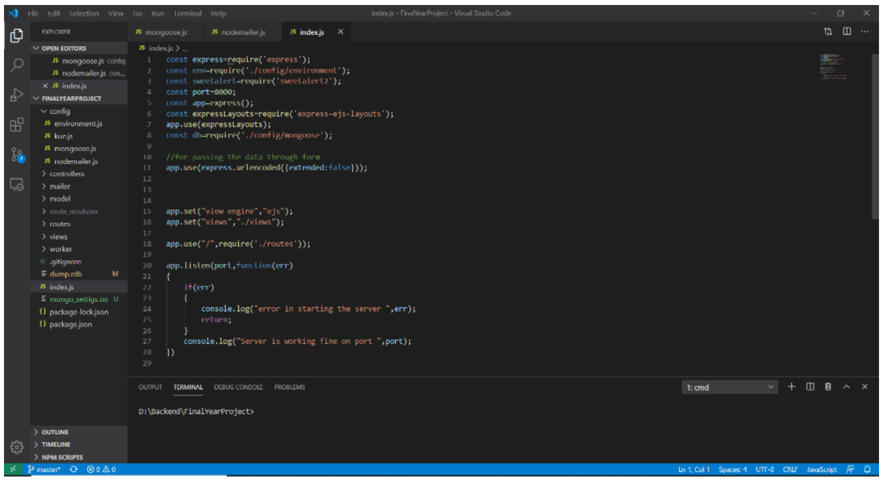
<u>IEEE Standards for WIFI Used:</u>



Feature	WiFi (802.11b)
PrimaryApplication	Wireless LAN
Frequency Band	2.4 GHz ISM
Channel Bandwidth	25 MHz
Half/Full Duplex	Half
Radio Technology	Direct Sequence Spread Spectrum
Bandwidth	<=0.44 bps/Hz
Efficiency	
Modulation	QPSK
FEC	None
Encryption	Optional- RC4m (AES in 802.11i)
Mobility	In development
Mesh	Vendor Proprietary
Access Protocol	CSMA/CA

Fig 4.5 WIFI IEEE Standards

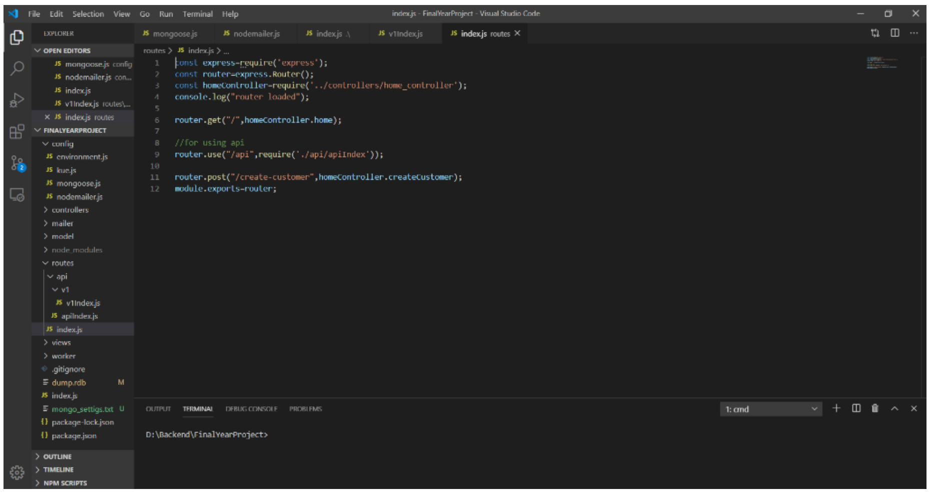
(a) Code for URL Deployment



```
const express=require('express');
const env=require('./config/environment');
const sweetalert=require('sweetalert2');
const port=8000;
const app=express();
const expressLayouts=require('express-ejs-layouts');
app.use(expressLayouts);
const db=require('./config/mongoose');

//for passing the data through form
app.use(express.urlencoded({extended:false}));



app.set("view engine","ejs");
app.set("views","./views");

app.use("/",require('./routes'));

app.listen(port,function(err)
{
    if(err)
    {
        console.log("error in starting the server ",err);
        return;
    }
    console.log("Server is working fine on port ",port);
})

```

```
D:\Backend\FinalYearProject>
```

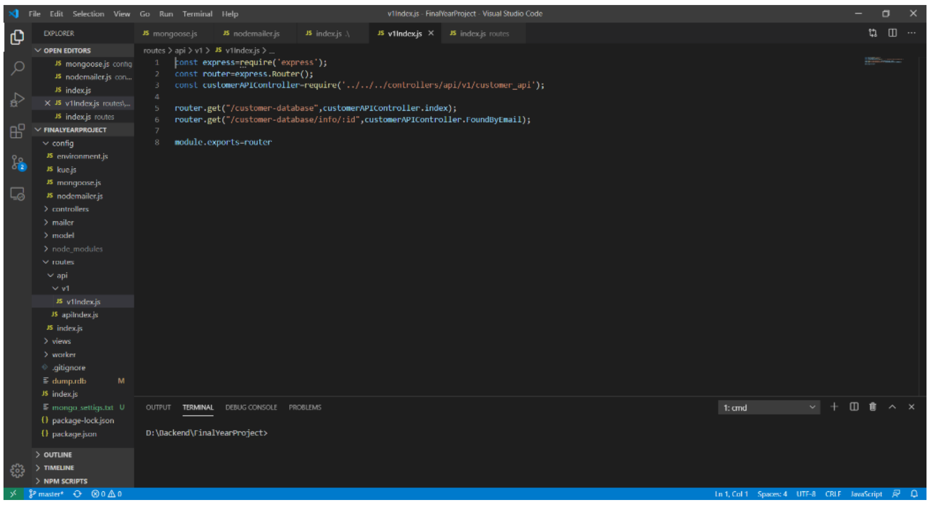
Fig 4.6 Code for URL Screenshot 1



```
const express=require('express');
const router=express.Router();
const homeController=require('../controllers/home_controller');
console.log("router loaded");

router.get("/",homeController.home);

//for using api
router.use("/api",require('./api/apiIndex'));

router.post("/create-customer",homeController.createCustomer);
module.exports=router;
```

```
D:\Backend\FinalYearProject>
```

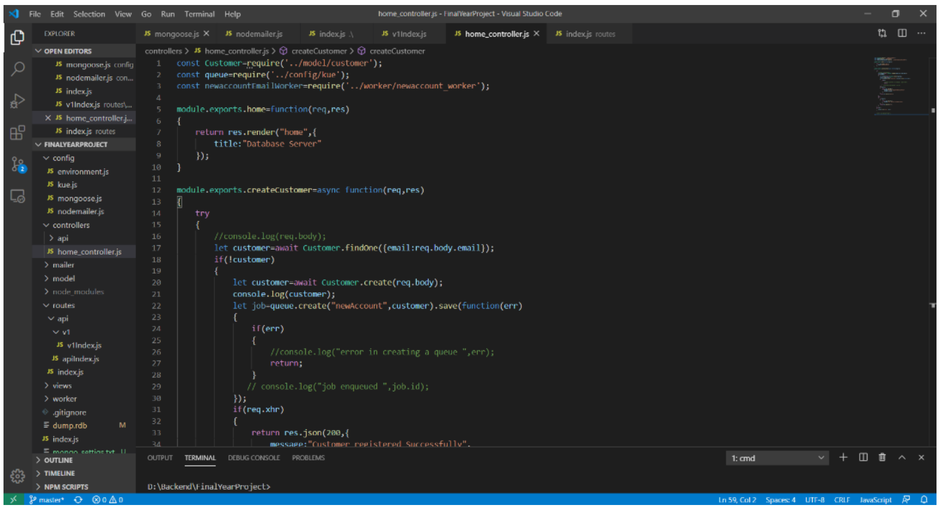
Fig 4.7 Code for URL Screenshot 2



```
const express=require('express');
const router=express.Router();
const customerAPIController=require('../../../controllers/api/v1/customer_api');

router.get("/customer-database",customerAPIController.index);
router.get("/customer-database/info/:id",customerAPIController.FoundByEmail);

module.exports=router
```

```
D:\Backend\FinalYearProject>
```

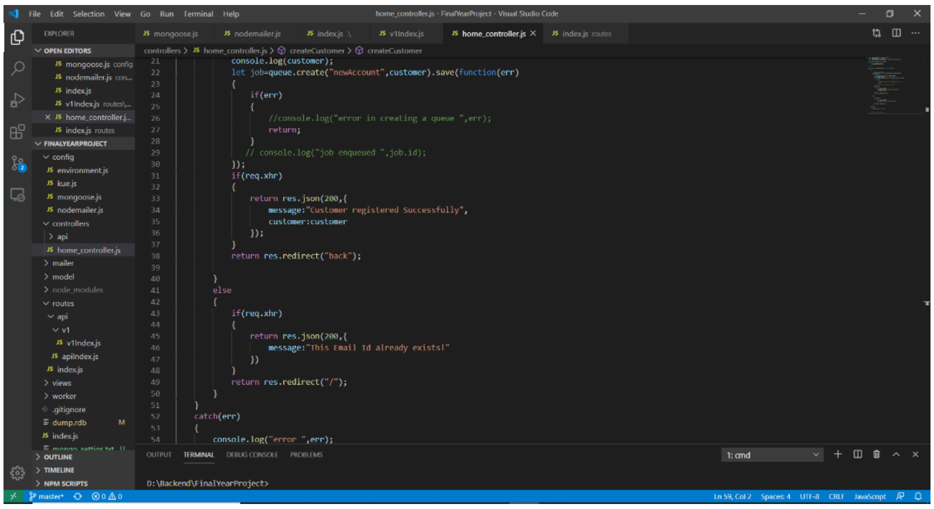
Fig 4.8 Code for URL Screenshot 3



```
const Customer=require('../model/customer');
const queue=require('../config/kue');
const newaccountEmailWorker=require('../worker/newaccount_worker');

module.exports.home=function(req,res)
{
    return res.render("home",{
        title:"Database Server"
    });
}

module.exports.createCustomer=async function(req,res)
{
    try
    {
        //console.log(req.body);
        let customer=await Customer.findOne({email:req.body.email});
        if(!customer)
        {
            let customer=await Customer.create(req.body);
            console.log(customer);
            let job=queue.create("newAccount",customer).save(function(err)
            {
                if(err)
                {
                    //console.log("error in creating a queue ",err);
                    return;
                }
               // console.log("job enqueued ",job.id);
            });
            if(req.xhr)
            {
                return res.json(200,{
                    message:"Customer registered Successfully",
```

```
D:\Backend\FinalYearProject>
```

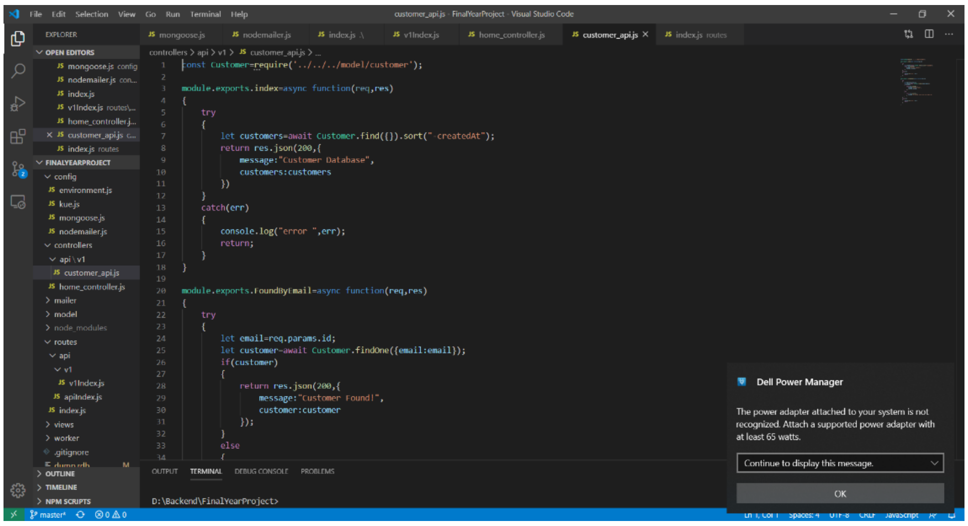
Fig 4.9 Code for URL Screenshot 4



```
        console.log(customer);
        let job=queue.create("newAccount",customer).save(function(err)
        {
            if(err)
            {
                //console.log("error in creating a queue ",err);
                return;
            }
           // console.log("job enqueued ",job.id);
        });
        if(req.xhr)
        {
            return res.json(200,{
                message:"Customer registered Successfully",
                customer:customer
            });
        }
        return res.redirect("back");

    }
    else
    {
        if(req.xhr)
        {
            return res.json(200,{
                message:"This Email Id already exists!"
            })
        }
        return res.redirect("/");
    }
}
catch(err)
{
    console.log("error ",err);
```

```
D:\Backend\FinalYearProject>
```

Fig 4.10 Code for URL Screenshot 5



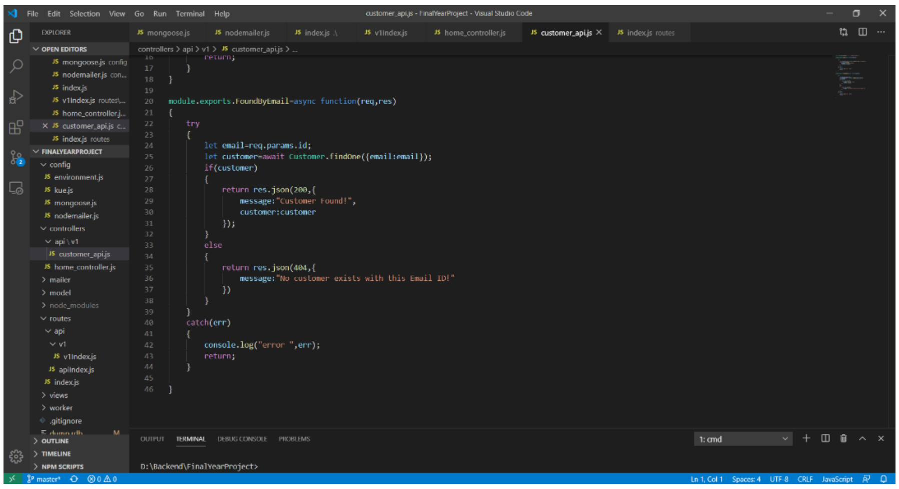
```
const Customer=require('../../../model/customer');

module.exports.index=async function(req,res)
{
    try
    {
        let customers=await Customer.find({}).sort("-createdAt");
        return res.json(200,{
            message:"Customer Database",
            customers:customers
        })
    }
    catch(err)
    {
        console.log("error ",err);
        return;
    }
}

module.exports.FoundByEmail=async function(req,res)
{
    try
    {
        let email=req.params.id;
        let customer=await Customer.findOne({email:email});
        if(customer)
        {
            return res.json(200,{
                message:"Customer Found!",
                customer:customer
            });
        }
        else
        {
```

```
D:\Backend\FinalYearProject>
```

Dell Power Manager

The power adapter attached to your system is not recognized. Attach a supported power adapter with at least 65 watts.

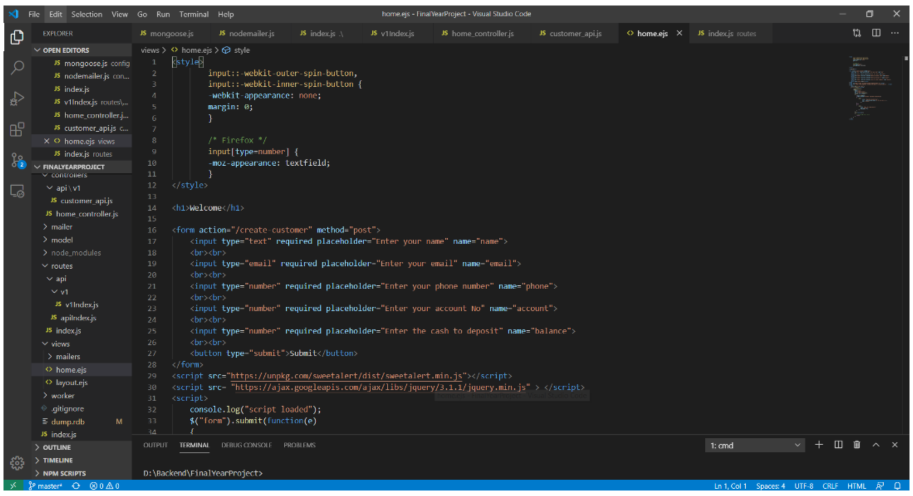
Fig 4.11 Code for URL Screenshot 6



```
    }
}

module.exports.FoundByEmail=async function(req,res)
{
    try
    {
        let email=req.params.id;
        let customer=await Customer.findOne({email:email});
        if(customer)
        {
            return res.json(200,{
                message:"Customer Found!",
                customer:customer
            });
        }
        else
        {
            return res.json(404,{
                message:"No customer exists with this Email ID!"
            })
        }
    }
    catch(err)
    {
        console.log("error ",err);
        return;
    }

}
```

```
D:\Backend\FinalYearProject>
```

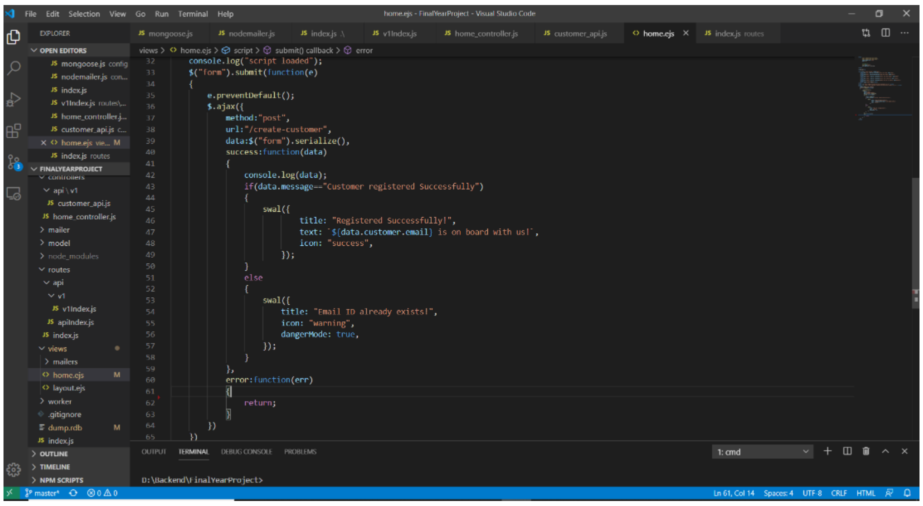
Fig 4.12 Code for URL Screenshot 7



```
<style>
        input::-webkit-outer-spin-button,
        input::-webkit-inner-spin-button {
        -webkit-appearance: none;
        margin: 0;
        }

        /* Firefox */
        input[type=number] {
        -moz-appearance: textfield;
        }
</style>

<h1>Welcome</h1>

<form action="/create-customer" method="post">
    <input type="text" required placeholder="Enter your name" name="name">
    <br><br>
    <input type="email" required placeholder="Enter your email" name="email">
    <br><br>
    <input type="number" required placeholder="Enter your phone number" name="phone">
    <br><br>
    <input type="number" required placeholder="Enter your account No" name="account">
    <br><br>
    <input type="number" required placeholder="Enter the cash to deposit" name="balance">
    <br><br>
    <button type="submit">Submit</button>
</form>
<script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
<script src= "https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js" > </script>
<script>
    console.log("script loaded");
    $("form").submit(function(e)
    {
```

```
D:\Backend\FinalYearProject>
```

Fig 4.13 Code for URL Screenshot 8

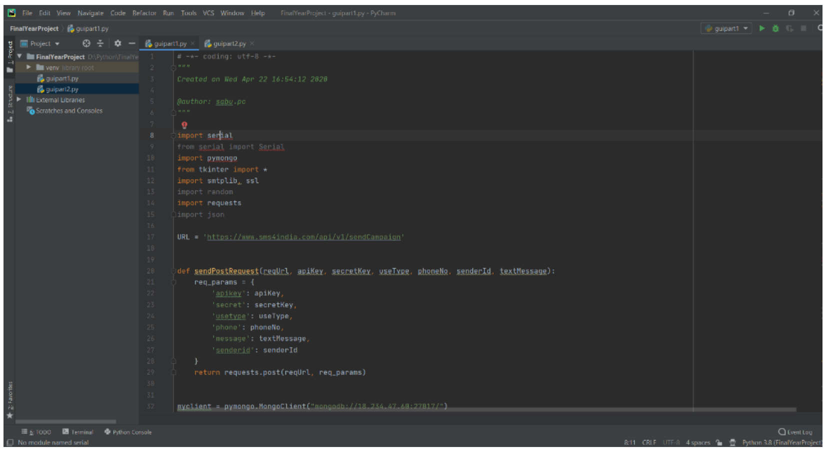


```
    console.log("script loaded");
    $("form").submit(function(e)
    {
        e.preventDefault();
        $.ajax({
            method:"post",
            url:"/create-customer",
            data:$("form").serialize(),
            success:function(data)
            {
                console.log(data);
                if(data.message=="Customer registered Successfully")
                {
                    swal({
                            title: "Registered Successfully!",
                            text: `${data.customer.email} is on board with us!`,
                            icon: "success",
                        });
                }
                else
                {
                    swal({
                        title: "Email ID already exists!",
                        icon: "warning",
                        dangerMode: true,
                    });
                }
            },
            error:function(err)
            {
                return;
            }
        })
    })
```

```
D:\Backend\FinalYearProject>
```

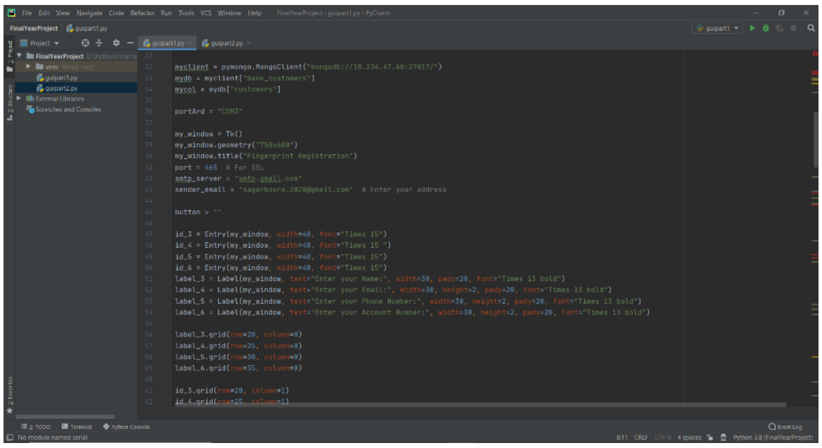
Fig 4.14 Code for URL Screenshot 9

(4) Code for Python Tkinter GUI Part 1



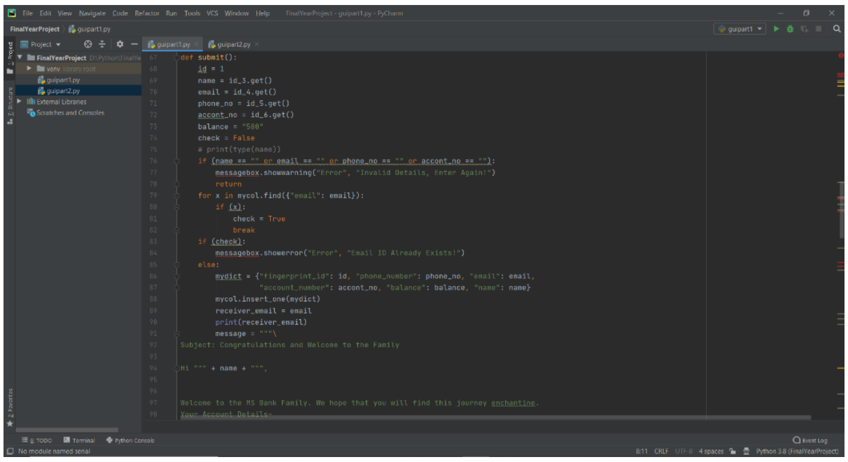
```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 22 16:54:12 2020

@author: sabu.pc
"""

import serial
from serial import Serial
import pymongo
from tkinter import *
import smtplib, ssl
import random
import requests
import json

URL = 'https://www.sms4india.com/api/v1/sendCampaign'


def sendPostRequest(reqUrl, apiKey, secretKey, useType, phoneNo, senderId, textMessage):
    req_params = {
        'apikey': apiKey,
        'secret': secretKey,
        'usetype': useType,
        'phone': phoneNo,
        'message': textMessage,
        'senderid': senderId
    }
    return requests.post(reqUrl, req_params)


myclient = pymongo.MongoClient("mongodb://18.234.47.68:27017/")
```

Fig 4.15 Code for GUI Part 1 Screenshot 1



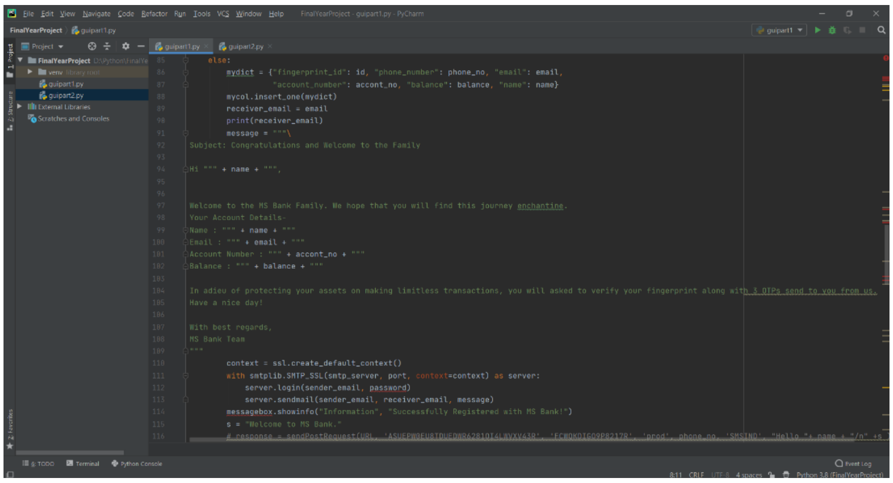
```python

myclient = pymongo.MongoClient("mongodb://18.234.47.60:27017/")
mydb = myclient["bank_customers"]
mycol = mydb["customers"]

portArd = "COM3"

my_window = Tk()
my_window.geometry("750x600")
my_window.title("Fingerprint Registration")
port = 465  # For SSL
smtp_server = "smtp.gmail.com"
sender_email = "sagarboora.2020@gmail.com"  # Enter your address

button = ""

id_3 = Entry(my_window, width=40, font="Times 15")
id_4 = Entry(my_window, width=40, font="Times 15 ")
id_5 = Entry(my_window, width=40, font="Times 15")
id_6 = Entry(my_window, width=40, font="Times 15")
label_3 = Label(my_window, text="Enter your Name:", width=30, pady=20, font="Times 13 bold")
label_4 = Label(my_window, text="Enter your Email:", width=30, height=2, pady=20, font="Times 13 bold")
label_5 = Label(my_window, text="Enter your Phone Number:", width=30, height=2, pady=20, font="Times 13 bold")
label_6 = Label(my_window, text="Enter your Account Number:", width=30, height=2, pady=20, font="Times 13 bold")

label_3.grid(row=20, column=0)
label_4.grid(row=25, column=0)
label_5.grid(row=30, column=0)
label_6.grid(row=35, column=0)

id_3.grid(row=20, column=1)
id_4.grid(row=25, column=1)
```

Fig 4.16 Code for GUI Part 1 Screenshot 2



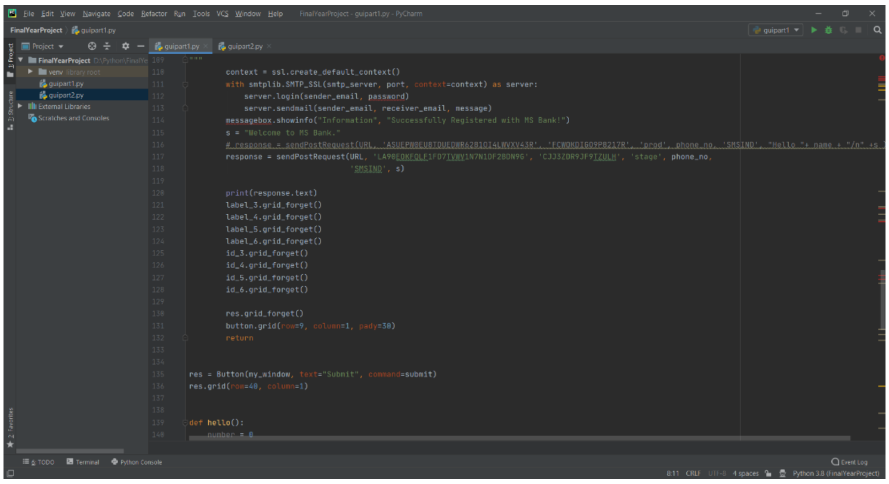
```python
def submit():
    id = 1
    name = id_3.get()
    email = id_4.get()
    phone_no = id_5.get()
    accont_no = id_6.get()
    balance = "500"
    check = False
    # print(type(name))
    if (name == "" or email == "" or phone_no == "" or accont_no == ""):
        messagebox.showwarning("Error", "Invalid Details, Enter Again!")
        return
    for x in mycol.find({"email": email}):
        if (x):
            check = True
            break
    if (check):
        messagebox.showerror("Error", "Email ID Already Exists!")
    else:
        mydict = {"fingerprint_id": id, "phone_number": phone_no, "email": email,
                  "account_number": accont_no, "balance": balance, "name": name}
        mycol.insert_one(mydict)
        receiver_email = email
        print(receiver_email)
        message = """\
Subject: Congratulations and Welcome to the Family

Hi """ + name + """,


Welcome to the MS Bank Family. We hope that you will find this journey enchantine.
Your Account Details-
```

Fig 4.17 Code for GUI Part 1 Screenshot 3



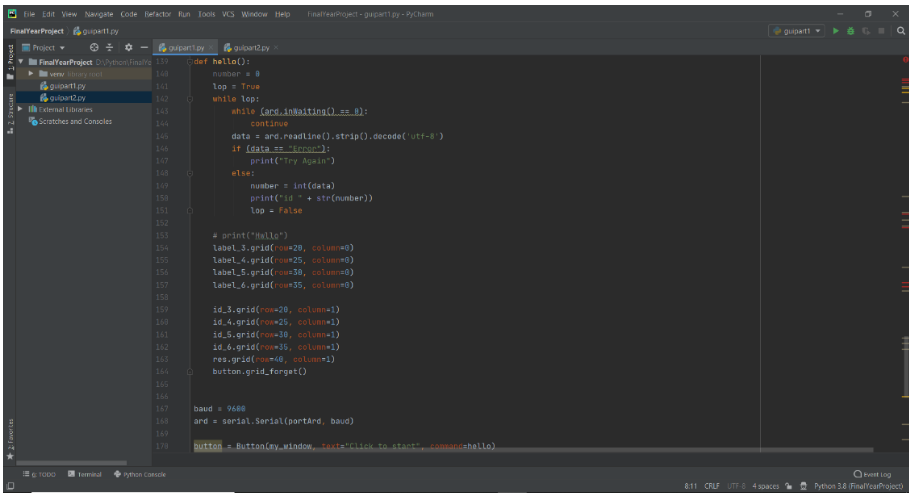
```python
    else:
        mydict = {"fingerprint_id": id, "phone_number": phone_no, "email": email,
                  "account_number": accont_no, "balance": balance, "name": name}
        mycol.insert_one(mydict)
        receiver_email = email
        print(receiver_email)
        message = """\
Subject: Congratulations and Welcome to the Family

Hi """ + name + """,


Welcome to the MS Bank Family. We hope that you will find this journey enchantine.
Your Account Details-
Name : """ + name + """
Email : """ + email + """
Account Number : """ + accont_no + """
Balance : """ + balance + """

In adieu of protecting your assets on making limitless transactions, you will asked to verify your fingerprint along with 3 OTPs send to you from us.
Have a nice day!

With best regards,
MS Bank Team
"""
        context = ssl.create_default_context()
        with smtplib.SMTP_SSL(smtp_server, port, context=context) as server:
            server.login(sender_email, password)
            server.sendmail(sender_email, receiver_email, message)
        messagebox.showinfo("Information", "Successfully Registered with MS Bank!")
        s = "Welcome to MS Bank."
        # response = sendPostRequest(URL, 'ASUEPW0EU8TDUEDWR62810I4LWVXV43R', 'FCWOKDIGO9P8217R', 'prod', phone_no, 'SMSIND', "Hello "+ name + "/n" +s
```

Fig 4.18 Code for GUI Part 1 Screenshot 4



```
"""
        context = ssl.create_default_context()
        with smtplib.SMTP_SSL(smtp_server, port, context=context) as server:
            server.login(sender_email, password)
            server.sendmail(sender_email, receiver_email, message)
        messagebox.showinfo("Information", "Successfully Registered with MS Bank!")
        s = "Welcome to MS Bank."
        # response = sendPostRequest(URL, 'ASUEPW0EU8TDUEDWR6281OI4LWVXV43R', 'FCWOKDIGO9P8217R', 'prod', phone_no, 'SMSIND', "Hello "+ name + "/n" +s
        response = sendPostRequest(URL, 'LA90EOKFOLF1FD7IVWV1N7N1DF2BDN9G', 'CJJ3ZDR9JF9IZULH', 'stage', phone_no,
                                   'SMSIND', s)

        print(response.text)
        label_3.grid_forget()
        label_4.grid_forget()
        label_5.grid_forget()
        label_6.grid_forget()
        id_3.grid_forget()
        id_4.grid_forget()
        id_5.grid_forget()
        id_6.grid_forget()

        res.grid_forget()
        button.grid(row=9, column=1, pady=30)
        return


res = Button(my_window, text="Submit", command=submit)
res.grid(row=40, column=1)


def hello():
    number = 0
```

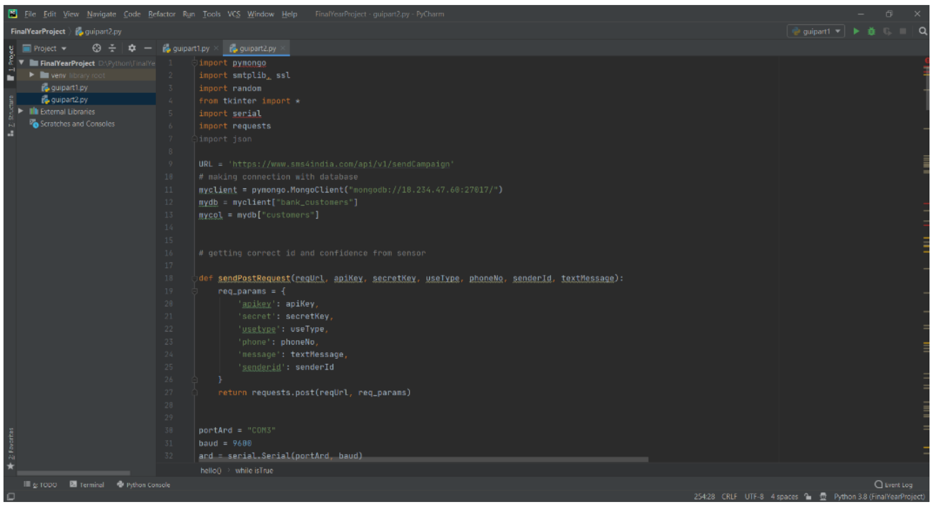
Fig 4.19 Code for GUI Part 1 Screenshot 5



```
def hello():
    number = 0
    lop = True
    while lop:
        while (ard.inWaiting() == 0):
            continue
        data = ard.readline().strip().decode('utf-8')
        if (data == "Error"):
            print("Try Again")
        else:
            number = int(data)
            print("id " + str(number))
            lop = False

    # print("Hwllo")
    label_3.grid(row=20, column=0)
    label_4.grid(row=25, column=0)
    label_5.grid(row=30, column=0)
    label_6.grid(row=35, column=0)

    id_3.grid(row=20, column=1)
    id_4.grid(row=25, column=1)
    id_5.grid(row=30, column=1)
    id_6.grid(row=35, column=1)
    res.grid(row=40, column=1)
    button.grid_forget()


baud = 9600
ard = serial.Serial(portArd, baud)

button = Button(my_window, text="Click to start", command=hello)
```

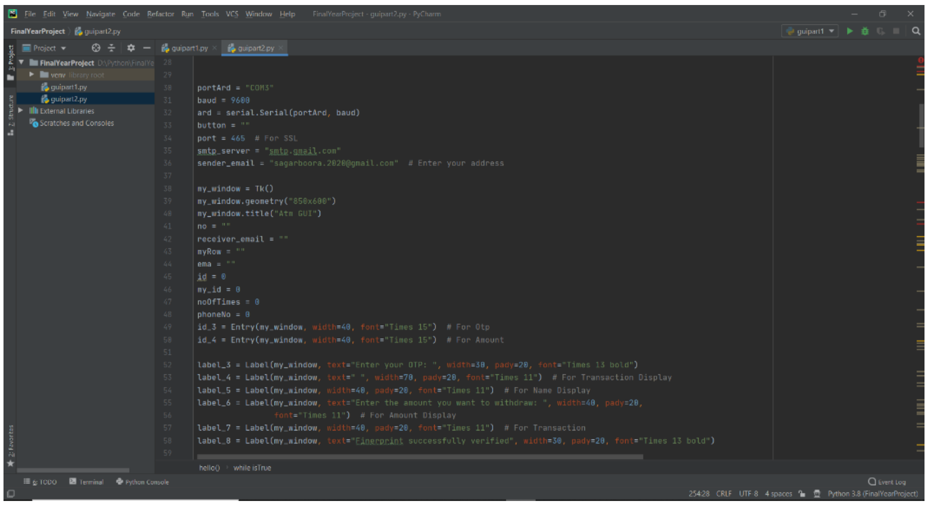
Fig 4.20 Code for GUI Part 1 Screenshot 6

(c) Code for Python Tkinter GUI Part 2



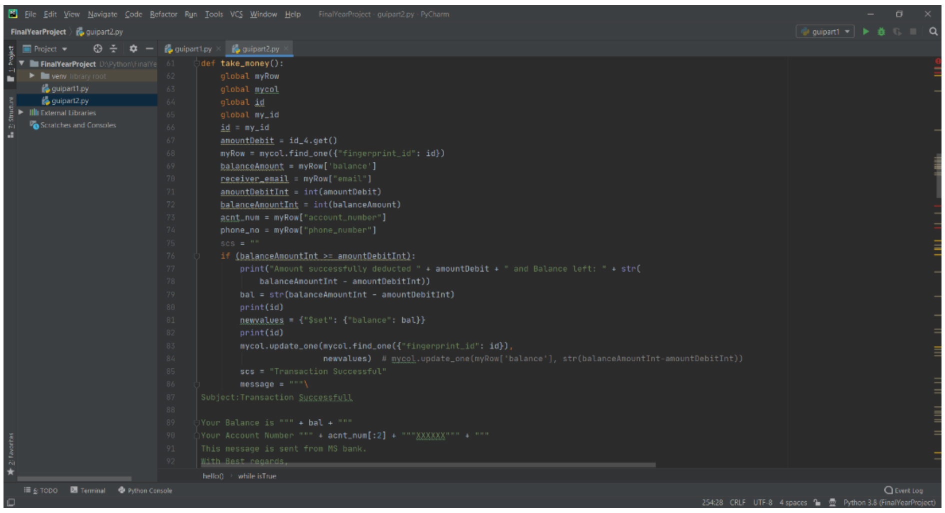
```
import pymongo
import smtplib, ssl
import random
from tkinter import *
import serial
import requests
import json

URL = 'https://www.sms4india.com/api/v1/sendCampaign'
# making connection with database
myclient = pymongo.MongoClient("mongodb://18.234.47.60:27017/")
mydb = myclient["bank_customers"]
mycol = mydb["customers"]


# getting correct id and confidence from sensor

def sendPostRequest(reqUrl, apiKey, secretKey, useType, phoneNo, senderId, textMessage):
    req_params = {
        'apikey': apiKey,
        'secret': secretKey,
        'usetype': useType,
        'phone': phoneNo,
        'message': textMessage,
        'senderid': senderId
    }
    return requests.post(reqUrl, req_params)


portArd = "COM3"
baud = 9600
ard = serial.Serial(portArd, baud)
```

Fig 4.21 Code for GUI Part 2 Screenshot 1



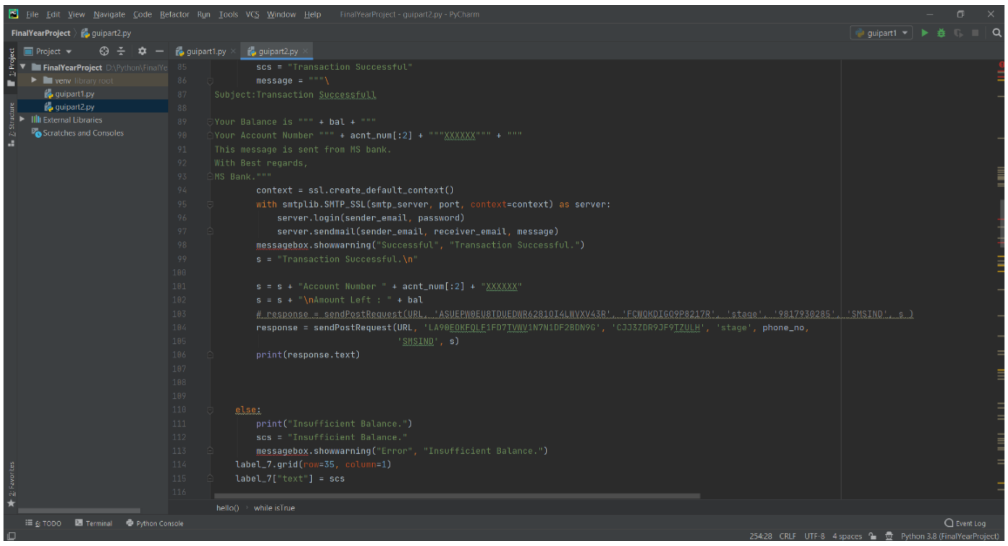
```python
portArd = "COM3"
baud = 9600
ard = serial.Serial(portArd, baud)
button = ""
port = 465  # For SSL
smtp_server = "smtp.gmail.com"
sender_email = "sagarboora.2020@gmail.com"  # Enter your address

my_window = Tk()
my_window.geometry("850x600")
my_window.title("Atm GUI")
no = ""
receiver_email = ""
myRow = ""
ema = ""
id = 0
my_id = 0
noOfTimes = 0
phoneNo = 0
id_3 = Entry(my_window, width=40, font="Times 15")  # For Otp
id_4 = Entry(my_window, width=40, font="Times 15")  # For Amount

label_3 = Label(my_window, text="Enter your OTP: ", width=30, pady=20, font="Times 13 bold")
label_4 = Label(my_window, text=" ", width=70, pady=20, font="Times 11")  # For Transaction Display
label_5 = Label(my_window, width=40, pady=20, font="Times 11")  # For Name Display
label_6 = Label(my_window, text="Enter the amount you want to withdraw: ", width=40, pady=20,
                font="Times 11")  # For Amount Display
label_7 = Label(my_window, width=40, pady=20, font="Times 11")  # For Transaction
label_8 = Label(my_window, text="Fingerprint successfully verified", width=30, pady=20, font="Times 13 bold")
```

Fig 4.22 Code for GUI Part 2 Screenshot 2



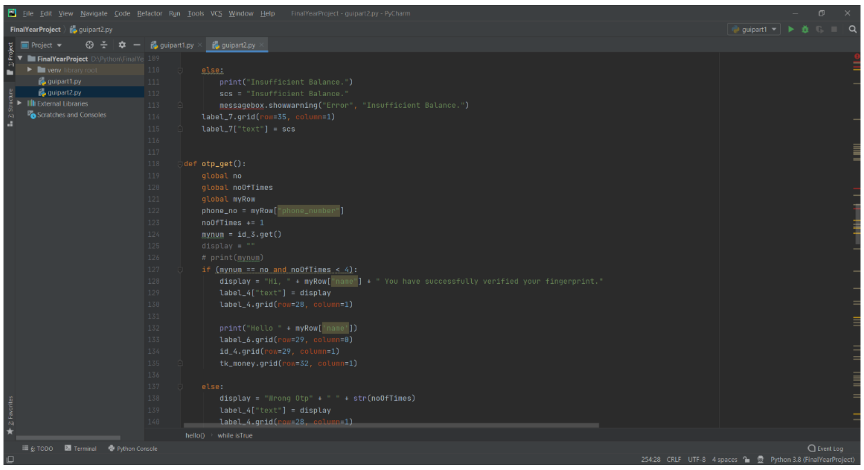
```python
def take_money():
    global myRow
    global mycol
    global id
    global my_id
    id = my_id
    amountDebit = id_4.get()
    myRow = mycol.find_one({"fingerprint_id": id})
    balanceAmount = myRow['balance']
    receiver_email = myRow["email"]
    amountDebitInt = int(amountDebit)
    balanceAmountInt = int(balanceAmount)
    acnt_num = myRow["account_number"]
    phone_no = myRow["phone_number"]
    scs = ""
    if (balanceAmountInt >= amountDebitInt):
        print("Amount successfully deducted " + amountDebit + " and Balance left: " + str(
            balanceAmountInt - amountDebitInt))
        bal = str(balanceAmountInt - amountDebitInt)
        print(id)
        newvalues = {"$set": {"balance": bal}}
        print(id)
        mycol.update_one(mycol.find_one({"fingerprint_id": id}),
                         newvalues)  # mycol.update_one(myRow['balance'], str(balanceAmountInt-amountDebitInt))
        scs = "Transaction Successful"
        message = """\
Subject:Transaction Successfull

Your Balance is """ + bal + """
Your Account Number """ + acnt_num[:2] + """XXXXXX""" + """
This message is sent from MS bank.
With Best regards,
```

Fig 4.23 Code for GUI Part 2 Screenshot 3



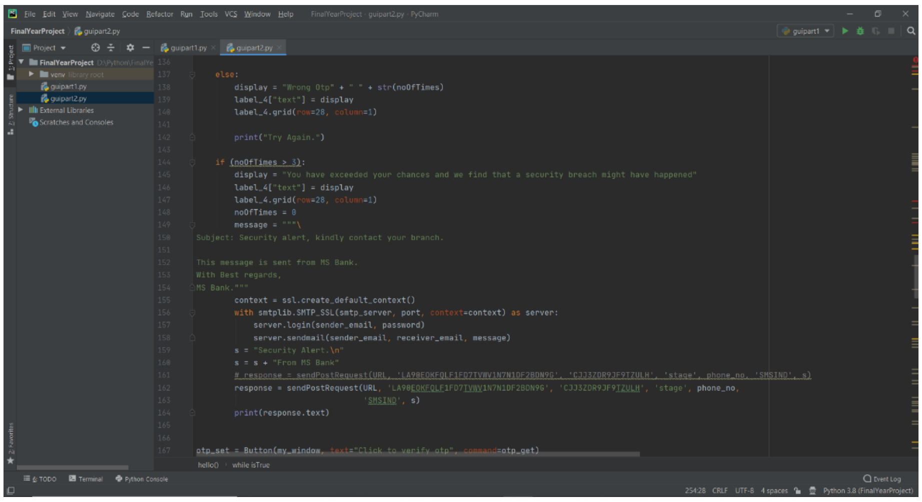
```python
        scs = "Transaction Successful"
        message = """\
Subject:Transaction Successfull

Your Balance is """ + bal + """
Your Account Number """ + acnt_num[:2] + """XXXXXX""" + """
This message is sent from MS bank.
With Best regards,
MS Bank."""
        context = ssl.create_default_context()
        with smtplib.SMTP_SSL(smtp_server, port, context=context) as server:
            server.login(sender_email, password)
            server.sendmail(sender_email, receiver_email, message)
        messagebox.showwarning("Successful", "Transaction Successful.")
        s = "Transaction Successful.\n"

        s = s + "Account Number " + acnt_num[:2] + "XXXXXX"
        s = s + "\nAmount Left : " + bal
        # response = sendPostRequest(URL, 'ASUEPW0EU8TDUEDWR6281OI4LWVXV43R', 'FCWQKDIGQ9P8217R', 'stage', '9817930285', 'SMSIND', s )
        response = sendPostRequest(URL, 'LA90EOKFQLF1FD7TVWV1N7N1DF2BDN9G', 'CJJ3ZDR9JF9TZULH', 'stage', phone_no,
                                   'SMSIND', s)
        print(response.text)



    else:
        print("Insufficient Balance.")
        scs = "Insufficient Balance."
        messagebox.showwarning("Error", "Insufficient Balance.")
    label_7.grid(row=35, column=1)
    label_7["text"] = scs
```

Fig 4.24 Code for GUI Part 2 Screenshot 4



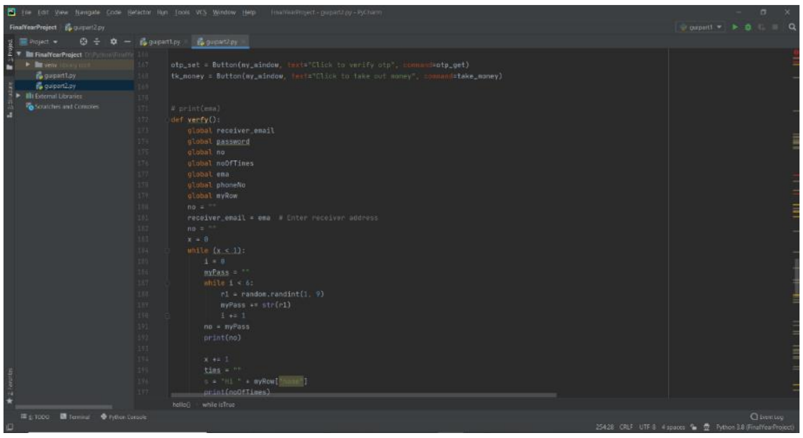
```
    else:
        print("Insufficient Balance.")
        scs = "Insufficient Balance."
        messagebox.showwarning("Error", "Insufficient Balance.")
    label_7.grid(row=35, column=1)
    label_7["text"] = scs


def otp_get():
    global no
    global noOfTimes
    global myRow
    phone_no = myRow["phone_number"]
    noOfTimes += 1
    mynum = id_3.get()
    display = ""
    # print(mynum)
    if (mynum == no and noOfTimes < 4):
        display = "Hi, " + myRow["name"] + " You have successfully verified your fingerprint."
        label_4["text"] = display
        label_4.grid(row=28, column=1)

        print("Hello " + myRow['name'])
        label_6.grid(row=29, column=0)
        id_4.grid(row=29, column=1)
        tk_money.grid(row=32, column=1)

    else:
        display = "Wrong Otp" + " " + str(noOfTimes)
        label_4["text"] = display
        label_4.grid(row=28, column=1)
```

Fig 4.25 Code for GUI Part 2 Screenshot 5



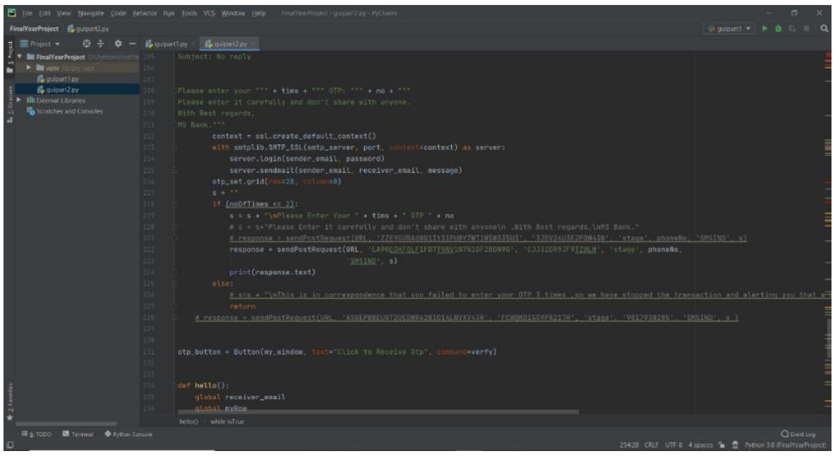
```
    else:
        display = "Wrong Otp" + " " + str(noOfTimes)
        label_4["text"] = display
        label_4.grid(row=28, column=1)

        print("Try Again.")

    if (noOfTimes > 3):
        display = "You have exceeded your chances and we find that a security breach might have happened"
        label_4["text"] = display
        label_4.grid(row=28, column=1)
        noOfTimes = 0
        message = """\
Subject: Security alert, kindly contact your branch.

This message is sent from MS Bank.
With Best regards,
MS Bank."""
        context = ssl.create_default_context()
        with smtplib.SMTP_SSL(smtp_server, port, context=context) as server:
            server.login(sender_email, password)
            server.sendmail(sender_email, receiver_email, message)
        s = "Security Alert.\n"
        s = s + "From MS Bank"
        # response = sendPostRequest(URL, 'LA90EOKFQLF1FD7TVWV1N7N1DF2BDN9G', 'CJJ3ZDR9JF9TZULH', 'stage', phone_no, 'SMSIND', s)
        response = sendPostRequest(URL, 'LA90EOKFQLF1FD7TVWV1N7N1DF2BDN9G', 'CJJ3ZDR9JF9TZULH', 'stage', phone_no,
                                   'SMSIND', s)
        print(response.text)


otp_set = Button(my_window, text="Click to verify otp", command=otp_get)
```

Fig 4.26 Code for GUI Part 2 Screenshot 6



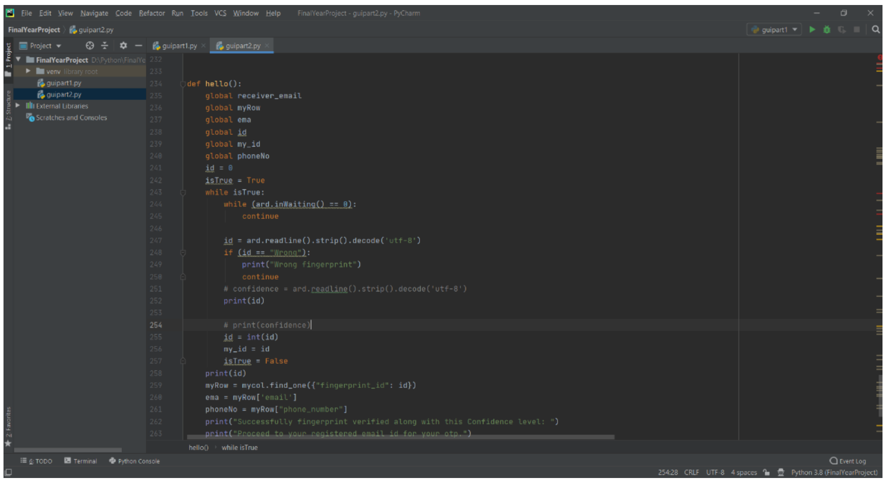
```
otp_set = Button(my_window, text="Click to verify otp", command=otp_get)
tk_money = Button(my_window, text="Click to take out money", command=take_money)


# print(ema)
def verfy():
    global receiver_email
    global password
    global no
    global noOfTimes
    global ema
    global phoneNo
    global myRow
    no = ""
    receiver_email = ema  # Enter receiver address
    no = ""
    x = 0
    while (x < 1):
        i = 0
        myPass = ""
        while i < 6:
            r1 = random.randint(1, 9)
            myPass += str(r1)
            i += 1
        no = myPass
        print(no)

        x += 1
        tims = ""
        s = "Hi " + myRow["name"]
        print(noOfTimes)
```

Fig 4.27 Code for GUI Part 2 Screenshot 7



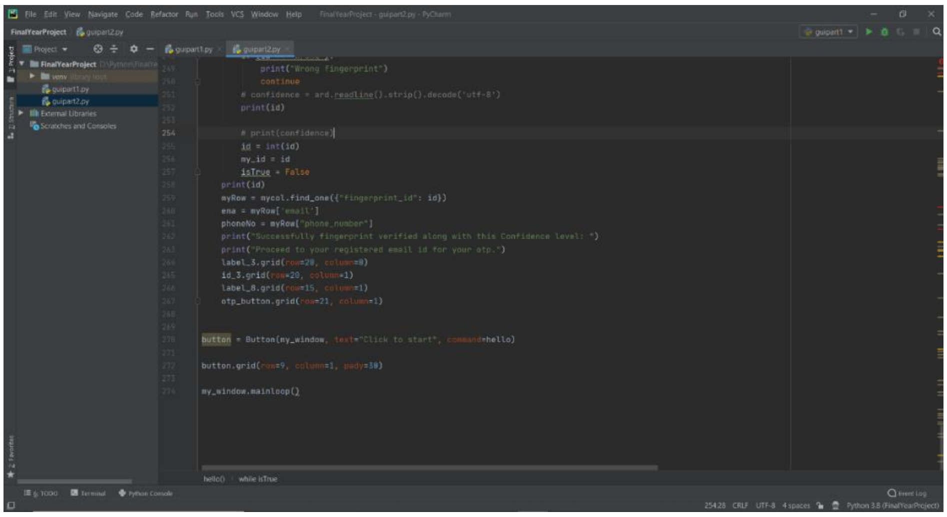
```
Subject: No reply


Please enter your """ + tims + """ OTP: """ + no + """
Please enter it carefully and don't share with anyone.
With Best regards,
MS Bank."""
        context = ssl.create_default_context()
        with smtplib.SMTP_SSL(smtp_server, port, context=context) as server:
            server.login(sender_email, password)
            server.sendmail(sender_email, receiver_email, message)
        otp_set.grid(row=28, column=0)
        s = ""
        if (noOfTimes <= 2):
            s = s + "\nPlease Enter Your " + tims + " OTP " + no
            # s = s+"Please Enter it carefully and don't share with anyone\n .With Best regards,\nMS Bank."
            # response = sendPostRequest(URL, '7ZF9GU8AU0U1IY31PU8Y7WT1WSW33SU3', '3JDV24U3F2FDN43W', 'stage', phoneNo, 'SMSIND', s)
            response = sendPostRequest(URL, 'LA90EDKFQLF1FD7IVNV1N7N1DF2BDN9G', 'CJJ3ZDR9JF9IZULH', 'stage', phoneNo,
                                       'SMSIND', s)
            print(response.text)
        else:
            # s=s + "\nThis is in correspondence that you failed to enter your OTP 3 times ,so we have stopped the transaction and alerting you that a
            return
    # response = sendPostRequest(URL, 'A5UEPW0EUSTDUEDWR62B1DI4LWVXY43R', 'FCWDKDIG09P8217R', 'stage', '9817938285', 'SMSIND', s )


otp_button = Button(my_window, text="Click to Receive Otp", command=verfy)


def hello():
    global receiver_email
    global myRow
```

Fig 4.28 Code for GUI Part 2 Screenshot 8



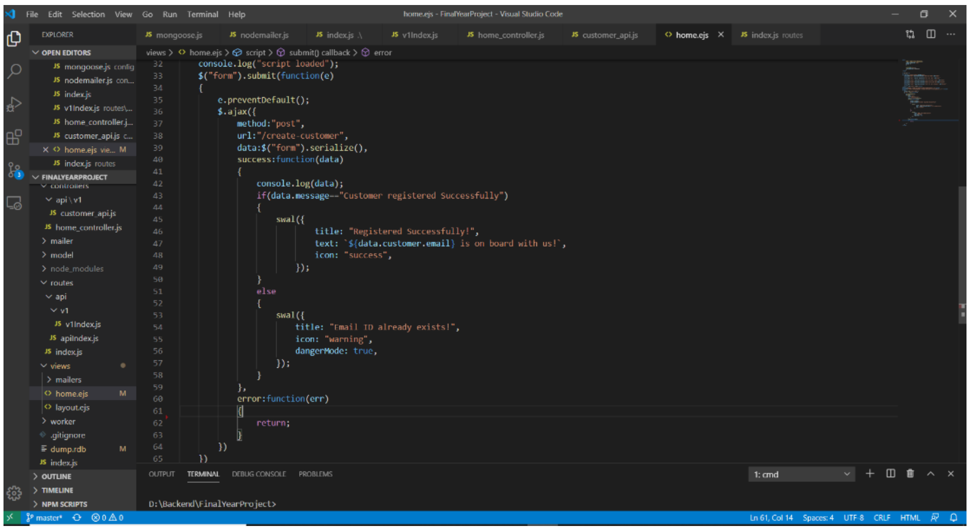
```


def hello():
    global receiver_email
    global myRow
    global ema
    global id
    global my_id
    global phoneNo
    id = 0
    isTrue = True
    while isTrue:
        while (ard.inWaiting() == 0):
            continue

        id = ard.readline().strip().decode('utf-8')
        if (id == "Wrong"):
            print("Wrong fingerprint")
            continue
        # confidence = ard.readline().strip().decode('utf-8')
        print(id)

        # print(confidence)
        id = int(id)
        my_id = id
        isTrue = False
    print(id)
    myRow = mycol.find_one({"fingerprint_id": id})
    ema = myRow['email']
    phoneNo = myRow["phone_number"]
    print("Successfully fingerprint verified along with this Confidence level: ")
    print("Proceed to your registered email id for your otp.")
```

Fig 4.29 Code for GUI Part 2 Screenshot 9



```
            print("Wrong fingerprint")
            continue
        # confidence = ard.readline().strip().decode('utf-8')
        print(id)

        # print(confidence)
        id = int(id)
        my_id = id
        isTrue = False
    print(id)
    myRow = mycol.find_one({"fingerprint_id": id})
    ema = myRow['email']
    phoneNo = myRow["phone_number"]
    print("Successfully fingerprint verified along with this Confidence level: ")
    print("Proceed to your registered email id for your otp.")
    label_3.grid(row=20, column=0)
    id_3.grid(row=20, column=1)
    label_8.grid(row=15, column=1)
    otp_button.grid(row=21, column=1)


button = Button(my_window, text="Click to start", command=hello)

button.grid(row=9, column=1, pady=30)

my_window.mainloop()
```

Fig 4.30 Code for GUI Part 2 Screenshot 10



```
    console.log("script loaded");
    $("form").submit(function(e)
    {
        e.preventDefault();
        $.ajax({
            method:"post",
            url:"/create-customer",
            data:$("form").serialize(),
            success:function(data)
            {
                console.log(data);
                if(data.message=="Customer registered Successfully")
                {
                    swal({
                            title: "Registered Successfully!",
                            text: `${data.customer.email} is on board with us!`,
                            icon: "success",
                        });
                }
                else
                {
                    swal({
                        title: "Email ID already exists!",
                        icon: "warning",
                        dangerMode: true,
                    });
                }
            },
            error:function(err)
            {
                return;
            }
        })
    })
```

```
D:\Backend\FinalYearProject>
```

Fig 4.31 Code for GUI Part 2 Screenshot 11

4.3 Constraints, Alternatives and Tradeoffs

The constraints that we come across when we implementing our code was that the GSM module was not working with the phone on the 4G network. It was only applicable to the 2G and 3G networks. Hence, this poses an obstacle to send text messages containing OTPs on the client's registered 4G/5G phone number. The alternative approach we come for this to use the python language for sending emails as well as text messages containing randomly generated passwords. The python is being linked with the Mongo DB. It runs a for-loop 6 times to generate a 6-digit arbitrary string of numbers each varying from 0 to 9.

Another constraint we faced during the deployment of our URL: www.msbank.co.in was that the Mongo DB was not secured as it should have been. It was not passcode protected. To handle this error, we reconfigure the mongod.conf on the AWS Ubuntu terminal. The reconfiguration consists of creating a newly authenticated user in the admin database.

Thus, these are the major constraints that we faced during the development of our research at the Vellore Institute of Technology, Vellore.

The alternatives and the tradeoffs that have been considered are Bluetooth Technology.

5 Schedule, Tasks and Milestones

Tab 5.1 Schedules

Schedule	Tasks

December 2019	During this period, we read online regarding the thefts happening around ATMs. One of the major conflicts was the cloning of the ATM cards and having the technology to predict the correct ATM PIN combination. **(Task 1)** The next step was to search for the optimized solution for the above-mentioned difficulty. Many research papers have come up with the idea of using Fingerprint Authentication as part of the security measures of ATM. But these research works have their limitations such as loss of the SIM card containing the OTP, GSM module not being operative on today's smartphones or being expensive for using other biometrics such as Palm, Iris, etc., **(Task 2)** Hence, we come up with the more suitable idea of using fingerprint authentication and sending out the OTPs to the phone as well as the Email. **(Task 3 Milestone)**
January 2020	After coming up with an idea, we have started working on our first stage of the research. We have started making the list of components required for this research. We did an online comparison of the various fingerprint sensors available on the market. We selected R3 07 because it was cost-effective and having a high rate of precision. **(Task 4 Milestone)** We established a structure following the careful analysis of the advantages and disadvantages of the components and how, to begin with, the code. **(Task 5)**
February 2020	As we proceed in February, our main objective was to study the algorithm behind the fingerprint authentication technology. After we thoroughly studied the algorithm using online resources. We have begun implementing the code for it in the C Language using Arduino Software. **(Task 6)** We were able to successfully implemented it in the third week of February. **(Task 7 Milestone)** Following that, we drafted the first three pages for our research paper. **(Task 8)**

March 2020	In the month of March, we started on the process to store data of our clients in our database. For that, we installed MongoDB. **(Task 9)** After that, we used python language as the medium between the clients and MongoDB. One of the advantages of the python language was that it can send out OTPs by extracting the accurate Email Address and Phone Number. It was only made possible after the thorough fingerprint authorization. **(Task 10 Milestone)** Thus, we assembled our components and tested each one of them with our code. Making the corrections in the code whenever required to optimize the execution of our project. We ended March by drafting three more pages of our research. **(Task 11)**
April 2020	During the last month of the research, we learned node language as it was a requirement for the deployment of our URL www.msbank.co.in. Generated APIs using this URL by importing the data from the Mongo DB. **(Task 12)** Rectifying minor errors in our codes and modifying it so that it could be connected with our URL. **(Task 13)** We also learned Python Tkinter to create the Graphical User Interface for the making of our ATM screen. After that, we internally connected both the codes. **(Task 14)** Finally, we completed the research paper with all the information and details. **(Task 15 Milestone)**

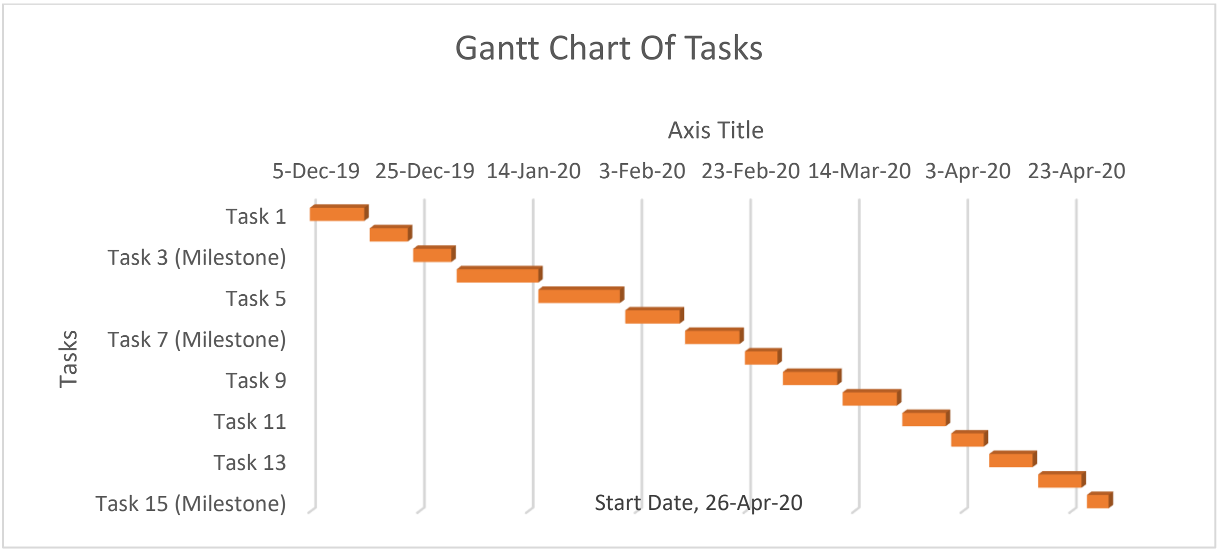


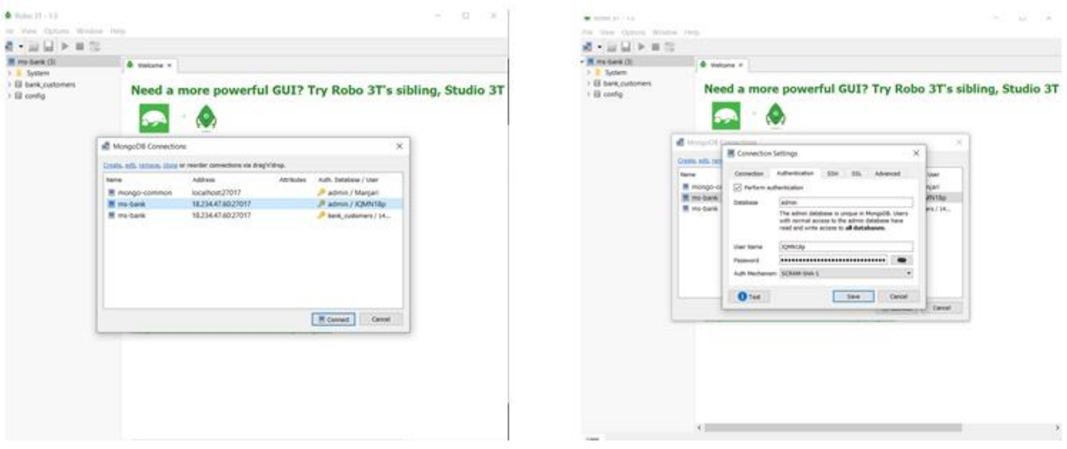
Fig 5.1 Gantt Chart of Tasks

6 Project Demonstration

The framework is introduced to implement explicit assignments, such as validating python code, email correlation, Phone compatibility, etc. and subsequently each method reset to start regulations.

The purposed methodology is different from the current ATM system as it incorporates fingerprint sensing in it. It will begin such as the user enters the ATM. It will be first labeled as to whether he/she is an existing customer or a new customer that wants to open their account in our bank. If he/she is an existing customer then they will be asked to directly proceed to the fingerprint authentication part of the process. Otherwise, he/she will be asked to register with us by submitting their frequently used Email Address and Phone Number along with his/her fingerprint in our database.

Here are the images of our secured database:



Create, edit, remove, clone or reorder connections via drag'n'drop.

Name	Address	Attributes	Auth. Database / User
mongo-common	localhost:27017		admin / Manjari
ms-bank	18.234.47.60:27017		admin / XQMN18p
ms-bank	18.234.47.60:27017		bank_customers / 14...

☑ Perform authentication

Database: admin

The admin database is unique in MongoDB. Users with normal access to the admin database have read and write access to **all databases.**

User Name: XQMN18p

Password: ••••••••••••••••••••••••••••••

Auth Mechanism: SCRAM-SHA-1

Fig 6.2 Password Protected DB

It shows that only one who has the authenticated credentials like admin and password has the ability to access the database.

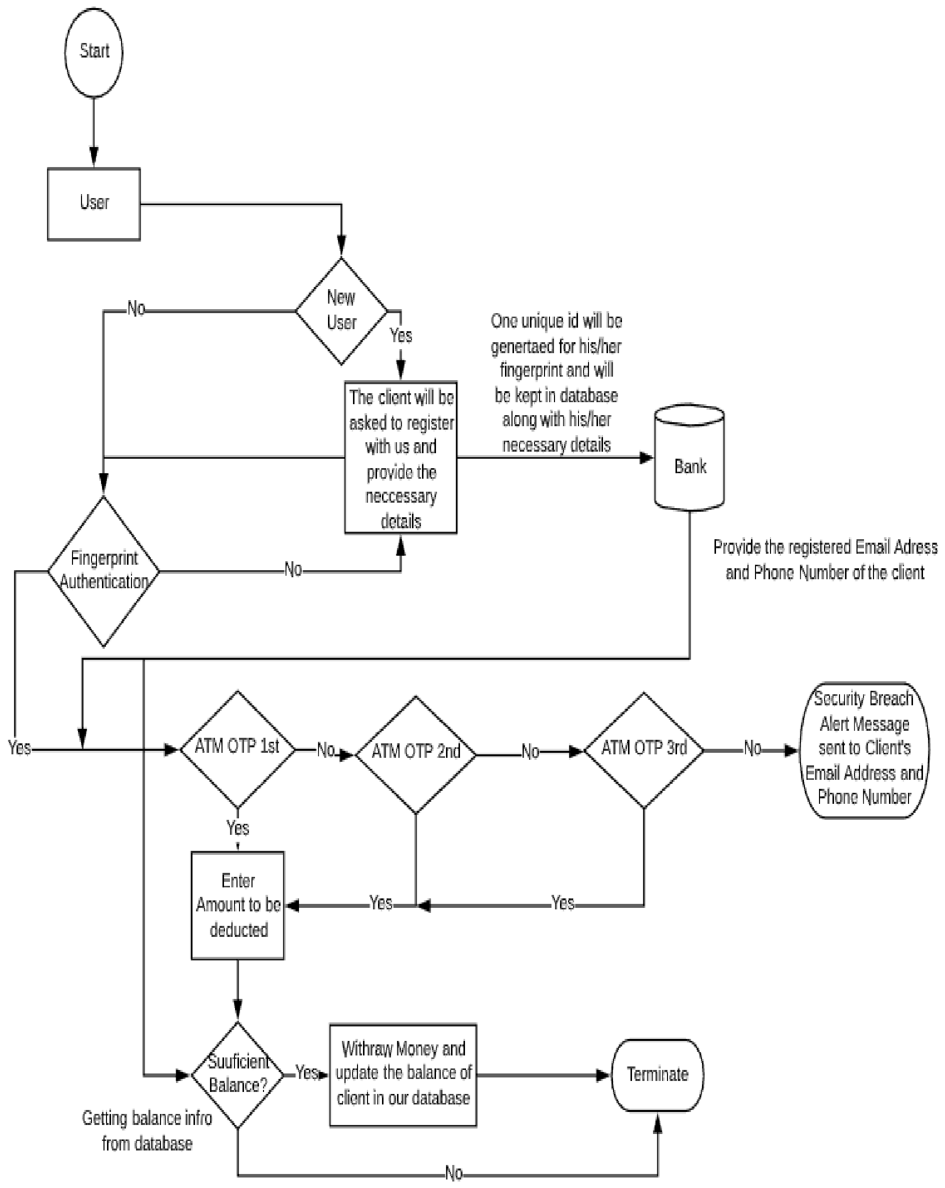
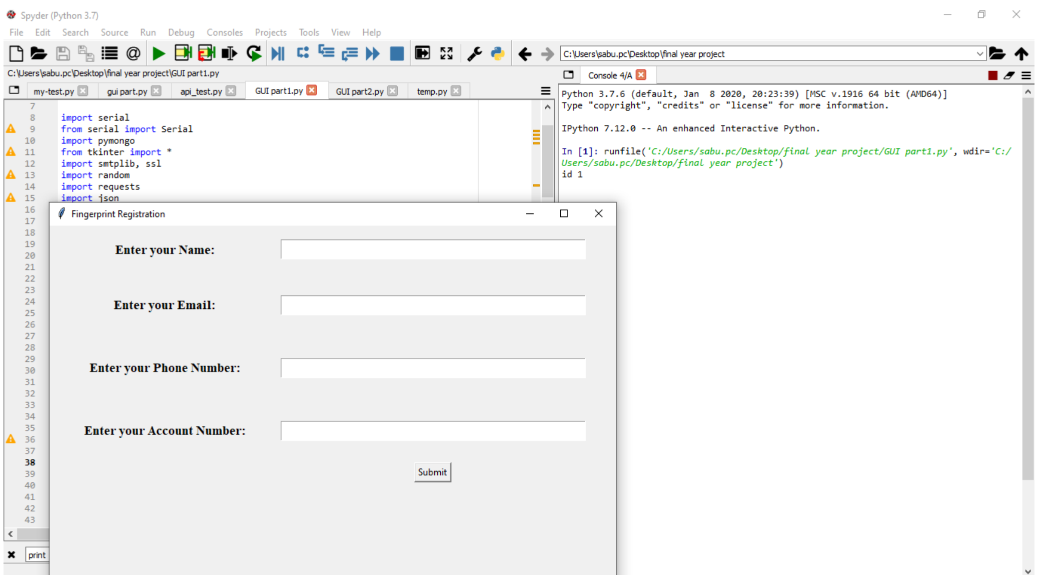

Start
User
New User
No
Yes
The client will be asked to register with us and provide the neccessary details
One unique id will be genertaed for his/her fingerprint and will be kept in database along with his/her necessary details
Bank
Fingerprint Authentication
No
Provide the registered Email Adress and Phone Number of the client
Yes
ATM OTP 1st
No
ATM OTP 2nd
No
ATM OTP 3rd
No
Security Breach Alert Message sent to Client's Email Address and Phone Number
Yes
Enter Amount to be deducted
Yes
Yes
Suuficient Balance?
Yes
Withraw Money and update the balance of client in our database
Terminate
Getting balance infro from database
No

Fig 6.1 Project Workflow

After that, both the existing and new customers will be acquired to give their fingerprints for verification of their existence in our database. They are allowed to scan their fingerprints as long as they reach a satisfying confidence match from our database. If the fingerprint authentication fails, then they will be restricted to take part in the next steps.

Thus, this is to ensure that the fraudulent activities happening as the hacker can easily trace the confidential information such as Card Number, Personal Information Security, and the mobile number will diminish to such an extent that it creates a sense of safe and secure environment.

On the successful verification of the fingerprint, they will be asked to enter the first OTP send to their registered Email Address and Phone Number. If they enter it correctly, they will be asked to enter the amount they want to withdraw. If the amount entered is less than the balance present in the bank account, they will successfully withdraw it. On the backend side, we will update the remaining balance in our database. If they fail to enter their first OTP accurately, then a second OTP will be sent to their Email Address and Phone Number again with following the same consequences. On the third OTP, they will have only one chance to enter it fairly. If they failed to do so, then from our side, a security breach mail and a text message will be sent to their registered Email Address and Phone Number and they will be locked out to proceed for the transaction of money. They have to start over with the whole process again. We are using Python GUI Tkinter as the interface to take the input from the user. We are using APIs and Mongo DB to extract and deliver the data of customers on the requirement. We have deployed an online URL for registering using WIFI with us: www.msbank.co.in. This URL is responsible for generating APIs.


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\sabu.pc\Desktop\final year project
C:\Users\sabu.pc\Desktop\final year project\GUI part1.py
my-test.py
gui part.py
api_test.py
GUI part1.py
GUI part2.py
temp.py
Console 4/A
Fingerprint Registration
Submit
print

```
import serial
from serial import Serial
import pymongo
from tkinter import *
import smtplib, ssl
import random
import requests
import json
```

```
Python 3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.12.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/sabu.pc/Desktop/final year project/GUI part1.py', wdir='C:/
Users/sabu.pc/Desktop/final year project')
id 1
```

Enter your Name:	
Enter your Email:	
Enter your Phone Number:	
Enter your Account Number:	

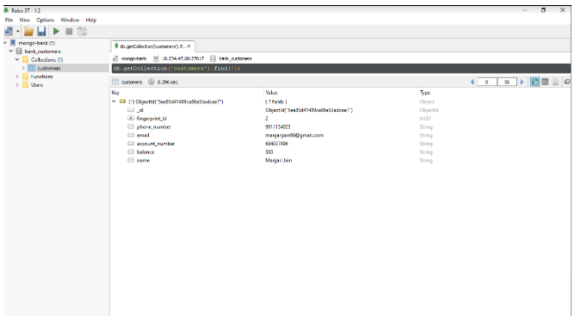
Fig 6.3 Python GUI for entering data.

It is the GUI for entering the details when the new client comes to the bank, it takes name, email, phone and account no as input.

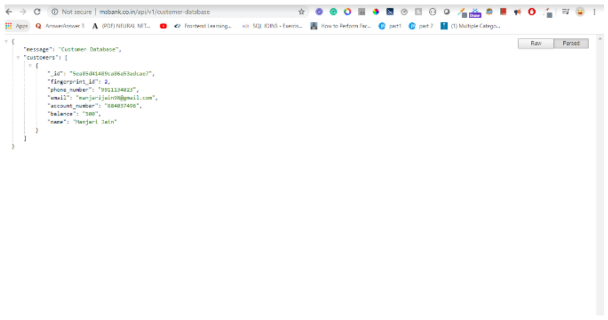
7 Cost Analysis/Result & Discussion

Here, we upload the technical input of our research work.

<u>Database:</u>



```
db.getCollection('customers').find({})
customers   0.296 sec.

Key                                        Value                                       Type
(1) ObjectId("5ea85d41489ca86a53adcae7")   { 7 fields }                                Object
    _id                                    ObjectId("5ea85d41489ca86a53adcae7")        ObjectId
    fingerprint_id                         2                                           Int32
    phone_number                           9911134023                                  String
    email                                  manjarijain98@gmail.com                     String
    account_number                         684037486                                   String
    balance                                500                                         String
    name                                   Manjari Jain                                String
```



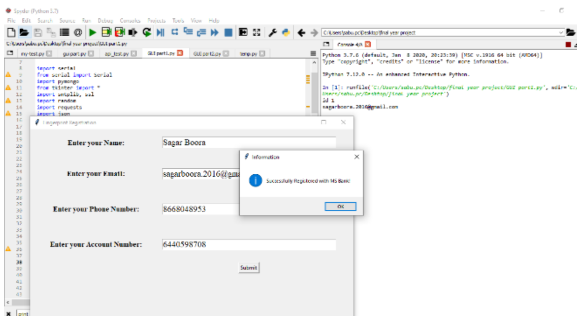
```
msbank.co.in/api/v1/customer-database
{
  "message": "Customer Database",
  "customers": [
    {
      "_id": "5ea85d41489ca86a53adcae7",
      "fingerprint_id": 2,
      "phone_number": "9911134023",
      "email": "manjarijain98@gmail.com",
      "account_number": "684037486",
      "balance": "500",
      "name": "Manjari Jain"
    }
  ]
}
```

(a) Mongo DB (b) API

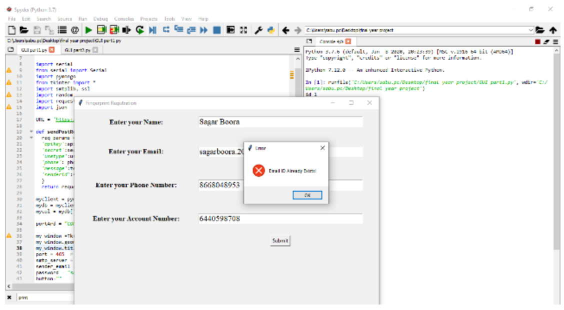
Fig 7.1.In the image a: Mongo DB (our original database) and image b: APIs

It shows our database of current customers registered in MS Bank

Storing new entry:



```
Enter your Name:            Sagar Boora
Enter your Email:           sagarboora.2016@gmail.com
Enter your Phone Number:    8668048953
Enter your Account Number:  6440598708
Submit

Information
Successfully Registered with MS Bank!
OK
```

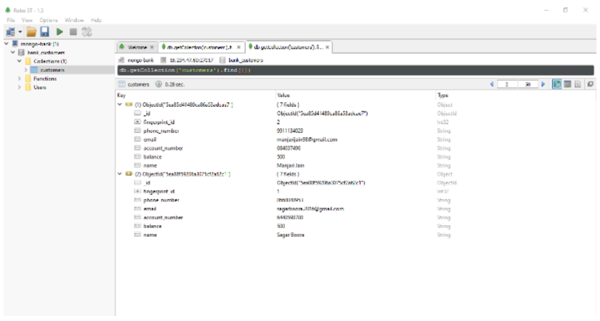


```
Enter your Name:            Sagar Boora
Enter your Email:           sagarboora.2016@gmail.com
Enter your Phone Number:    8668048953
Enter your Account Number:  6440598708
Submit

Error
Email ID Already Exists!
OK
```

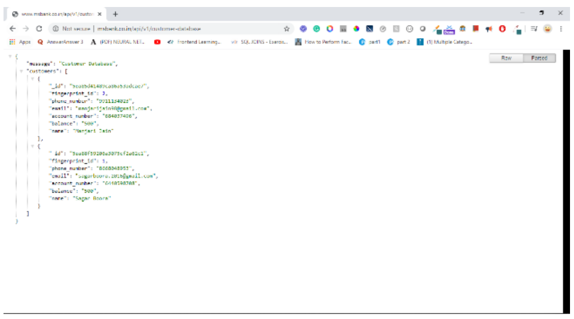
(a) Storing new data (b) Warning for using the same email address

Fig 7.2. In Image (a), the new user enters the bank, this GUI appears on the ATM Screen which stores his fingerprint and personal information such as Name, Email, Phone Number, and Account Number. After entering the details, a successful message pop up will appear. In Image (b), It shows that the same email address can not be used again as it already exists in our database.

Updation of current database with a new entry:



```
db.getCollection('customers').find({})
customers   0.28 sec.

Key                                        Value                                       Type
(1) ObjectId("5ea85d41489ca86a53adcae7")   { 7 fields }                                Object
    _id                                    ObjectId("5ea85d41489ca86a53adcae7")        ObjectId
    fingerprint_id                         2                                           Int32
    phone_number                           9911134023                                  String
    email                                  manjarijain98@gmail.com                     String
    account_number                         684037486                                   String
    balance                                500                                         String
    name                                   Manjari Jain                                String
(2) ObjectId("5ea88f59208a3075cf2a82c1")   { 7 fields }                                Object
    _id                                    ObjectId("5ea88f59208a3075cf2a82c1")        ObjectId
    fingerprint_id                         1                                           Int32
    phone_number                           8668048953                                  String
    email                                  sagarboora.2016@gmail.com                   String
    account_number                         6440598708                                  String
    balance                                500                                         String
    name                                   Sagar Boora                                 String
```

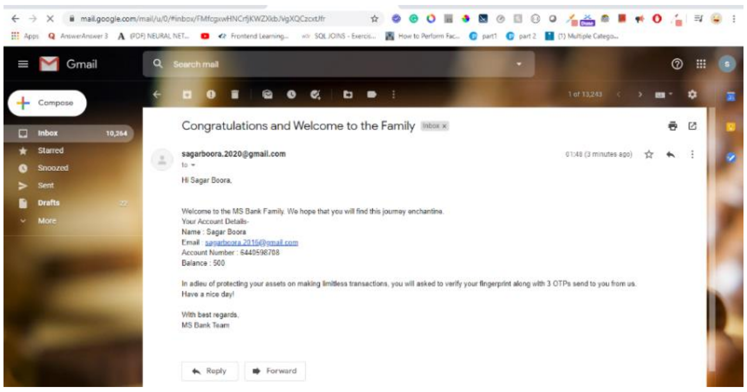


```
msbank.co.in/api/v1/customer-database
{
  "message": "Customer Database",
  "customers": [
    {
      "_id": "5ea85d41489ca86a53adcae7",
      "fingerprint_id": 2,
      "phone_number": "9911134023",
      "email": "manjarijain98@gmail.com",
      "account_number": "684037486",
      "balance": "500",
      "name": "Manjari Jain"
    },
    {
      "_id": "5ea88f59208a3075cf2a82c1",
      "fingerprint_id": 1,
      "phone_number": "8668048953",
      "email": "sagarboora.2016@gmail.com",
      "account_number": "6440598708",
      "balance": "500",
      "name": "Sagar Boora"
    }
  ]
}
```

(a) Mongo DB Updated (b) API Updated

Fig 7.3. In Image a: Mongo DB and image b: APIs are updated with the database of the new user. It shows the new data has been successfully stored in our database as well as APIs

<u>Confirmation mail sent:</u>


Congratulations and Welcome to the Family Inbox x

sagarboora.2020@gmail.com
to
01:48 (3 minutes ago)

Hi Sagar Boora,

Welcome to the MS Bank Family. We hope that you will find this journey enchantine.
Your Account Details-
Name : Sagar Boora
Email : sagarboora.2016@gmail.com
Account Number : 6440598708
Balance : 500

In adieu of protecting your assets on making limitless transactions, you will asked to verify your fingerprint along with 3 OTPs send to you from us.
Have a nice day!

With best regards,
MS Bank Team

Fig 7.4. Confirmation Mail. (After successful registration, MS Bank sent a confirmation mail to the new user along with his/her account details that he/she enters in figure 7.2 (a)).

After successful fingerprint verification, two cases arise such as:

<u>CASE 1: Case of entering OTP correctly:</u>

It represents that any registered customer, who enters the OTP matching from the text messages and emails, is allowed to make the transactions depending on the amount present in his/her bank account.

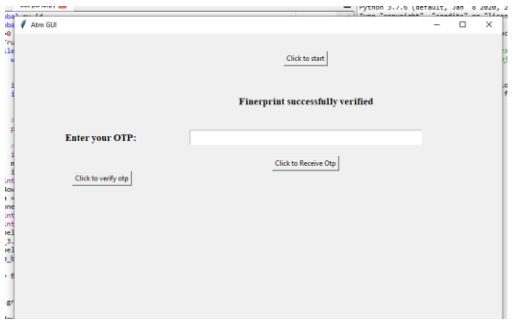
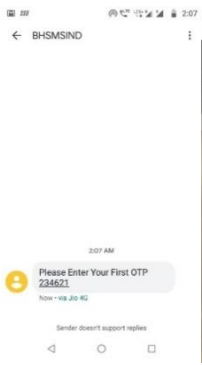


Fig 7.5. Successful Fingerprint Verification. (After successful fingerprint verification of the newly registered user, then a new GUI appears asking for the first OTP to be entered here.)

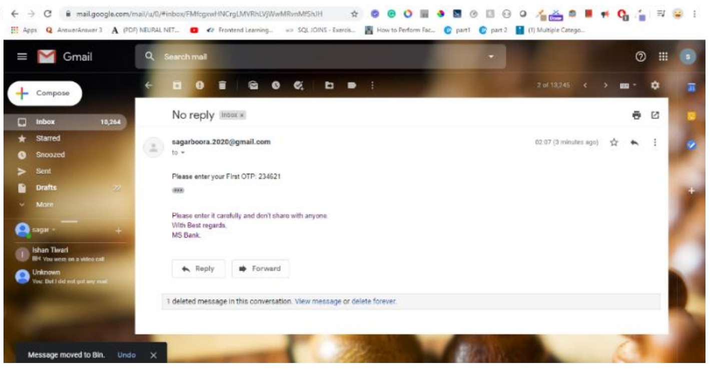

BHSMSIND

2:07 AM

Please Enter Your First OTP 234621

Now • via Jio 4G

Sender doesn't support replies


No reply Inbox x

sagarboora.2020@gmail.com
to
02:07 (3 minutes ago)

Please enter your First OTP: 234621

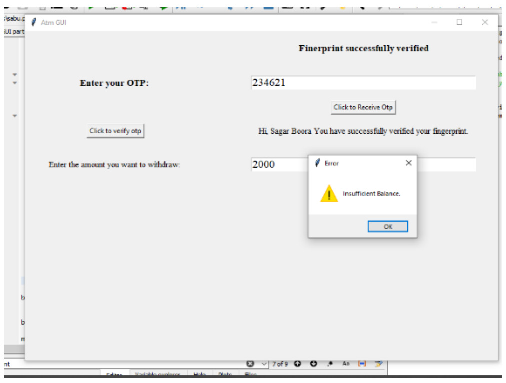
Please enter it carefully and don't share with anyone.
With Best regards,
MS Bank.

1 deleted message in this conversation. View message or delete forever.

Message moved to Bin. Undo

(a) C1: First OTP on Phone (b) C1: First OTP on Email

Fig 7.6. The same first OTP was sent on the new user's registered Phone Number (Image a) and Email Address (Image b). This OTP is required to be entered by the new user after his/her successful fingerprint authentication


Atm GUI

Finerprint successfully verified

Enter your OTP: 234621

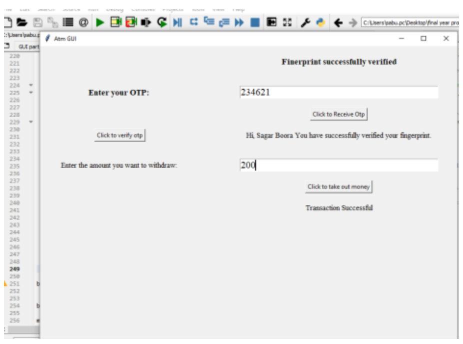
Hi, Sagar Boora You have successfully verified your fingerprint.

Enter the amount you want to withdraw: 2000

Error

Insufficient Balance.

Fig 7.7. C1: Correct OTP with insufficient balance. When the customer enters the correct OTP but put the insufficient amount to be dispensed then it shows a warning of having "Insufficient Balance" This GUI reflects that the client doesn't have the authority to withdraw the amount when the money present in his/her account is less than the amount entered.


Atm GUI

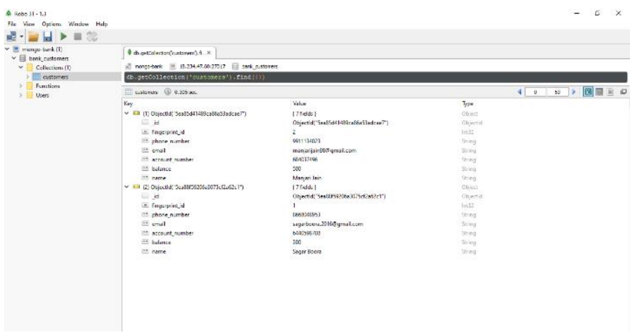
Finerprint successfully verified

Enter your OTP: 234621

Hi, Sagar Boora You have successfully verified your fingerprint.

Enter the amount you want to withdraw: 200

Transaction Successful



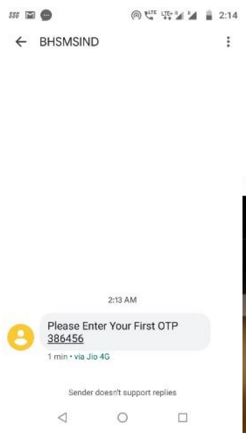
```
mongo-bank   18.234.47.60:27017   bank_customers
db.getCollection('customers').find({})
customers   0.305 sec.
Key                                              Value                                        Type
(1) ObjectId("5ea85d41489ca86a53adcae7")         { 7 fields }                                 Object
    _id                                          ObjectId("5ea85d41489ca86a53adcae7")         ObjectId
    fingerprint_id                               2                                            Int32
    phone_number                                 9911134023                                   String
    email                                        manjarijain98@gmail.com                      String
    account_number                               684037496                                    String
    balance                                      500                                          String
    name                                         Manjari Jain                                 String
(2) ObjectId("5ea88f59206a3075cf2a62c1")         { 7 fields }                                 Object
    _id                                          ObjectId("5ea88f59206a3075cf2a62c1")         ObjectId
    fingerprint_id                               1                                            Int32
    phone_number                                 8860340953                                   String
    email                                        sagarboora.2016@gmail.com                    String
    account_number                               6440598708                                   String
    balance                                      300                                          String
    name                                         Sagar Boora                                  String
```

(a) C1: Correct OTP with sufficient balance (b) C1: Remaining balance updated in Mongo DB

Fig 7.8. In Image (a), when the customer enters a significant amount (less than the money present in his/her account) then it is allowed to be dispersed and in the image (b), we update the remaining amount in our database.

<u>CASE 2: Case of entering OTP incorrectly:</u>

This is the case where any registered customer fails to enter the OTP accurately for the third time. Due to this, they will receive a "Security alert, kindly contact your nearest branch as soon as possible" Email.


2:14

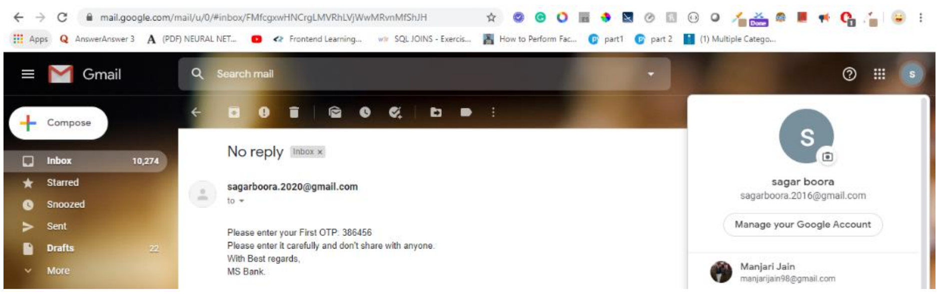
BHSMSIND

2:13 AM

Please Enter Your First OTP 386456

1 min • via Jio 4G

Sender doesn't support replies



No reply Inbox

sagarboora.2020@gmail.com

to

Please enter your First OTP: 386456
Please enter it carefully and don't share with anyone.
With Best regards,
MS Bank.

(a) C2: First OTP on Phone (b) C2: First OTP on Email

Fig 7.9. The same first OTP was sent on the user's registered Phone Number (Image a) and Email Address (Image b). This OTP is required to be entered by the user after his/her successful fingerprint authentication.

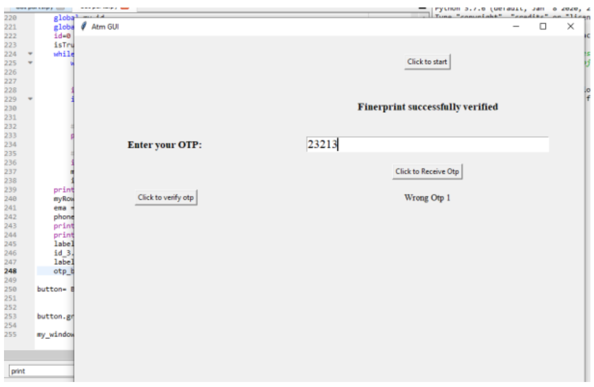
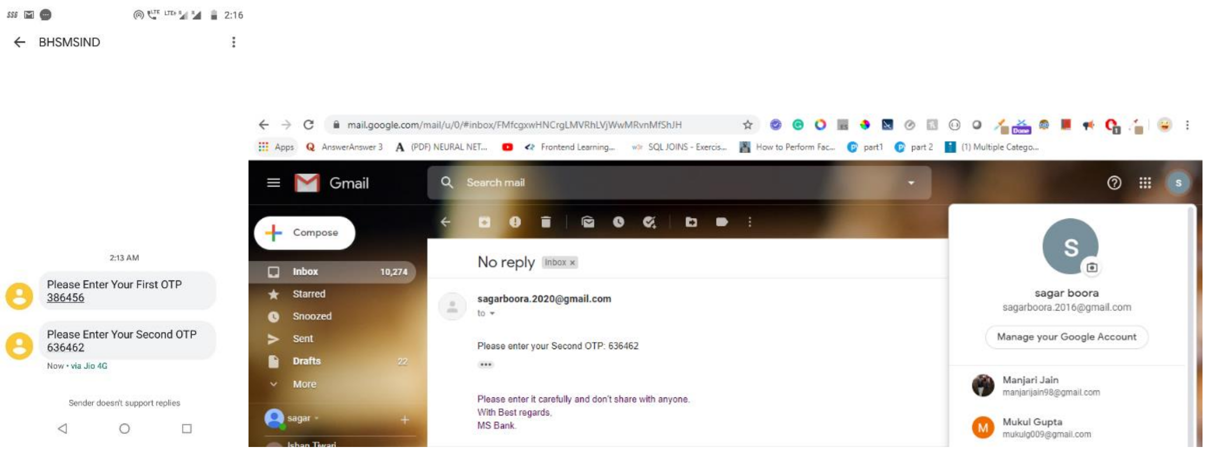


Fig 7.10. C2: First OTP wrong (The customer enters the first OTP wrong.(When the customer enters the first OTP which doesn't match the one send to the registered Email and Phone No, then the GUI prints a warning stating that "Wrong otp 1". After this customer has to click "Click to Receive OTP" in to receive the second OTP.)


BHSMSIND

2:13 AM

Please Enter Your First OTP 386456

Please Enter Your Second OTP 636462

Now • via Jio 4G

Sender doesn't support replies

No reply

sagarboora.2020@gmail.com

to

Please enter your Second OTP: 636462

Please enter it carefully and don't share with anyone.
With Best regards,
MS Bank.

(a) C2: Second OTP on Phone (b) C2: Second OTP on Email

Fig 7.11. The same second OTP was sent on the user's registered Phone Number (Image a) and Email Address (Image b). This OTP is required to be entered by the user after his/her successful fingerprint authentication.

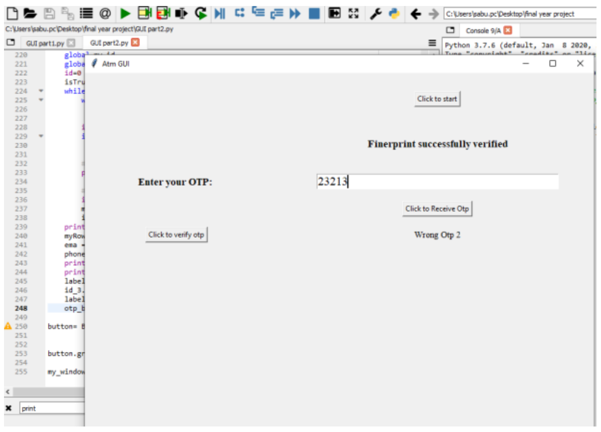
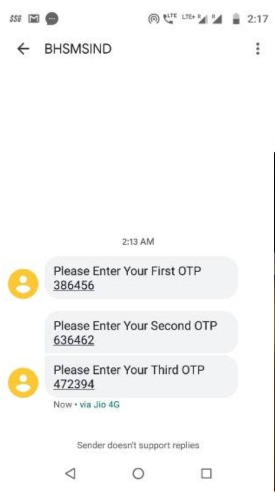


Fig 7.12. C2: First OTP wrong. (The customer enters the second OTP wrong. When the customer enters the first OTP which doesn't match the one send to the registered Email and Phone No, then the GUI prints a warning stating that "Wrong otp 1". After this customer has to click "Click to Receive OTP" in to receive the third OTP.)



BHSMSIND

2:13 AM

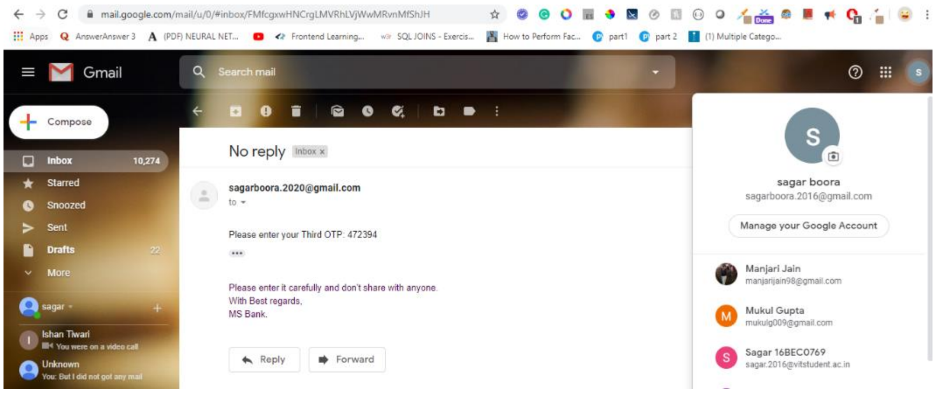
Please Enter Your First OTP 386456

Please Enter Your Second OTP 636462

Please Enter Your Third OTP 472394

Now • via Jio 4G

Sender doesn't support replies



No reply

sagarboora.2020@gmail.com

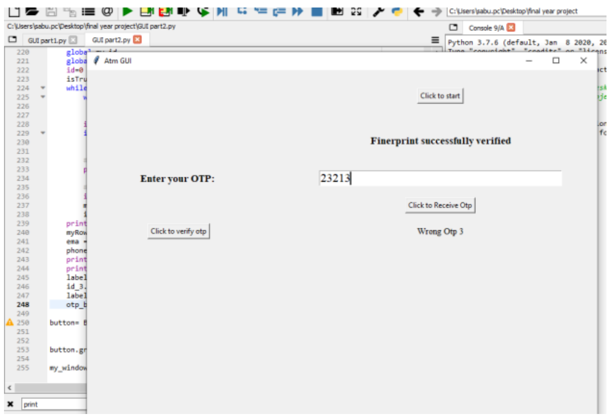
to

Please enter your Third OTP: 472394

Please enter it carefully and don't share with anyone.
With Best regards,
MS Bank.

(a) C2: Third OTP on Phone (b) C2: Third OTP on Email

Fig 7.13. The same third OTP was sent on the user's registered Phone Number (Image a) and Email Address (Image b). This OTP is required to be entered by the user after his/her successful fingerprint authentication.



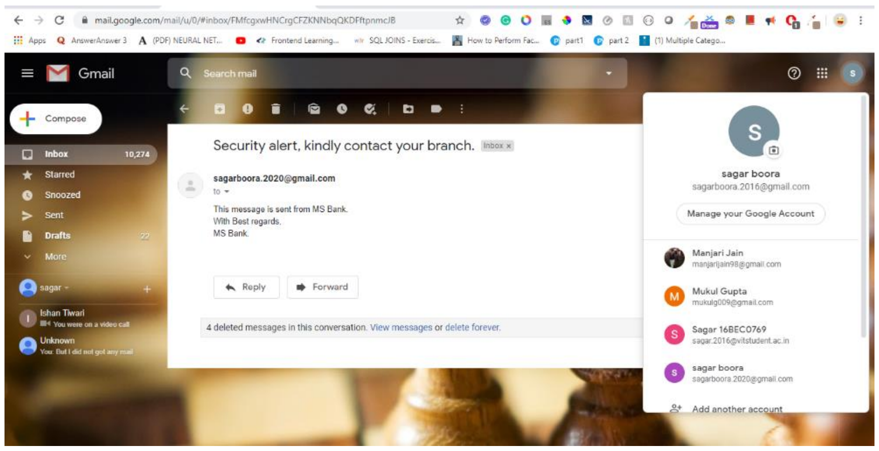
Atm GUI

Finerprint successfully verified

Enter your OTP: 23213

Wrong Otp 3

Fig 7.14. Third OTP Wrong. (The customer enters the third OTP wrong. When the customer enters the second OTP which doesn't match the one send to the registered Email and Phone No, then the GUI prints a warning stating that "Wrong otp 3". After this, the customer is not allowed to make a transaction and has to start the process again to withdraw money. Along with this restriction, he/she receives an alert email from our side.)



Security alert, kindly contact your branch.

sagarboora.2020@gmail.com

to

This message is sent from MS Bank.
With Best regards,
MS Bank.

4 deleted messages in this conversation. View messages or delete forever.

Fig 7.15. Security Alert Message. (Security Theft Alert Message to the customer's registered Email Address after he/she failed at the third time to enter the OTP accurately.)

8 Summary

There are some discrepancies in the past research for improving the ATM security using fingerprint authentication as being discussed in the Literature Synopsis. Thus, there was a necessity to improvise the scope of ATM safety with a better and cost-effective solution. The answer is to incorporate fingerprints as well but the direction of sending OTPs varies from the previous studies, as OTPs are conveyed out in two ways: client’s registered email address and phone number. The methodology works such that it provides the alternative to the new client to register with the bank using Python GUI (ATM Screen). For the registered clients, the ATM Fingerprint Sensor first acknowledges that the biometric is authenticated. After that, an OTP is forwarded to the contact details of the client. If the client fails to enter the first one correctly, he/she is left with 2 more possibilities to enter it accurately. On the account of third OTP failure, the client (or suspected hacker) is blocked from making any transactions and a Security Theft Alert mail is also dispatched. A URL www.msbank.co.in is also utilized for the online registrations of clients as well as constructing APIs to safely store the data of our new and enrolled clients. As a result, in case of WIFI absence, the client can use his/her mobile OTP, or in case of SIM lost, the customer can retrieve that same OTP from his/her logged email address. In conclusion, the results obtained from the research proved that this theory is attainable.

9 References

1. Kavita Hooda, International Journal of Scientific and Research Publications, Volume 6, Issue 4, April 2016. “ATM Security”, pp-1.

2. Mithun Dutta, Kangkhita Keam Psyche and Shamima Yasmin, American Journal of Engineering Research (AJER), Volume 6, Issue 8, August 2017, “ATM Transaction Security Using Fingerprint Recognition”, pp-41-45.

3. Zhao, F., Tang, X., The Journal of the Pattern Recognition Society, Volume 40, Issue 4, April 2007, “Preprocessing and postprocessing for skeleton-based fingerprint minutiae extraction”, pp-1270-1281.

4. F. A. Afsar, M. Arif and M. Hussain, National Conference on Emerging Technologies, 2004, “Fingerprint Identification and Verification System using Minutiae Matching,”, pp.141-146.

5. Hmadri nath moulick , Joyjit Patra, Arun kanti Manna, International Journal of Engineering Inventions, Volume 3, Issue 1, August 2013, “Computer Assisted and Contour Detectioin in Medical ImagingUsing Fuzzy Logic”, pp-1-7.

6. Frimpong Twum, Kofi Nti, Michael Asante, International Journal of Science and Engineering Applications, Volume 5, Issue 3, May 2016, “Improving Security Levels In Automatic Teller Machines (ATM) Using Multifactor Authentication”, pp-503.

7. Sanjay, S. G., International Journal Of Engineering And Computer Science, 2014, “ATM Transaction Security System Using Biometric Palm Print Recognition and Transaction Confirmation System”, pp-5332-5335.

8. Lalzirtira, National Institute of Technology Rourkela Mtech Thesis, 2013, “Graphical User Authentication, India".

9. Anand, D. A., Dinesh, G. & Naveen, H. D., International Jouranl of Communication and Computer Technologies (IJCCT), 2013. “A Reliable ATM Protocol and Comparative Analysis on Various Parameters with other ATM Protocols”, pp. 192-197.

10. Saropourian, B., ICCSIT 2009, 2nd IEEE International Conference, 2009, “A new approach of finger-print recognition based on neural network” , pp. 158-161.

11. Ratha, N., Connell, J. & Bolle, R., IBM Systems Journal, Volume 40, No. 3, 2001, “Enhancing Security and Privacy in Biometrics-based Authentication Systems”, pp. 614-634.

12. Sowmya Ravikumar, Sandhya Vaidyanathan, B. Thamotharan, S. Ramakrishnan, International Journal of Engineering and Technology, Volume 5, No. 3, 2013, “A new business model for ATM transaction security using fingerprint recognition”.

13. Petrlic, Ronald, and Christoph Sorge, IET Information Security, Volume 8, Issue 2, 2013, “Establishing user trust in automated teller machine integrity”, pp-132-139.

14. Myo, N., International Conference on Education Technology and Computer, 2009, “Fingerprint Identification Based on the Model of the Outer Layers of Polygon Subtraction”, pp-201 – 204.

15. Journal for Research, Volume 2, Issue 12, February 2017.

16. Ms. Archana S. Shinde and Prof. Varsha Bendre, 2015 International Conference on Computing Communication Control and Automation, 2015, “An Embedded Fingerprint Authentication System”, pp-45.

17. Jun Zhou, Guangda Sua, Chun hongJiang, Neurocomputing, 70 , 2007, “A face and fingerprint identity authentication system based on multiroute detection” pp-922-931.

18. Yuliang He, Jie Tian, Xiping Luo, Tanghui Zhang, Pattern Recognition, Letters 24, 2003, “Image enhancement and minutiae matching in fingerprint verification”, pp-1349-1360.

19. Wei Wang, Jianwei Li, Feifei Huang, Hailiang Feng, Pattern Recognition, Letters 29, 2008, “Design and implementation of Log-Gabor filter in fingerprint image enhancement”, pp-301-308.

20. Lin Hong, Wan Yifei, Anil Jain, IEEE Transactions on Pattern Analysis and Machine intelligence, Volume 20, Issue 8, 1998, “Fingerprint image enhancement: algorithm and performance evaluation”, pp- 777-789.

21. Der Chin Chen, Biometric Systems, Design and Applications, 2011, “Portable Biometric System of High Sensitivity Absorption Detection”,.

22. Subhra Mazumdar, Venkata Dhulipala, San Diego, “Biometric Security Using Finger Print Recognition”.

23. S.M. Shamsheer Daula, Dr.K.E Sreenivasa Murthy, International Journal of Advanced and Innovative Research, Volume 1, Issue 2, July 2012, “An Embedded ATM Security Design using ARM Processor with Fingerprint recognition and GSM”.

24. Steve Furber, ARM System-on-Chip Architecture, Second Edition, 2000, ISBN 0-201-67519-6.

25. Anil K. Jain, Jianjiang Feng, Karthik Nandakumar, IEEE Computer Society 2010, 0018-9162/10, “Fingerprint Matching”, pp. 36-44.