“Voice Enabled ATM”

A Socially Relevant Project-I Report submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
ANANTAPUR.

In Partial Fulfillment of the Requirements for the Award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERINGBY

G. Naga Eshitha	**19121A0565**
C. Anusha	**19121A0538**
Abdul Bari Shaik	**19121A0502**
G. Hemanth Kumar	**19121A0558**
B. Nagendra Naik	**19121A0524**

Under the Guidance of
Dr. K. Reddy Madhavi

Associate Professor,
Department of Computer Science and Engineering

Department of Computer Science and Engineering
SREE VIDYANIKETHAN ENGINEERING COLLEGE
(Affiliated to JNTUA, Anantapuramu)
Sree Sainath Nagar, Tirupathi – 517 1022019-2023

SREE VIDYANIKETHANENGINEERINGCOLLEGE

(Affiliated to Jawaharlal Nehru Technological University Anantapur)
Sree Sainath Nagar, A. Rangampet, Tirupati – 517 102, Chittoor Dist., A.P.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Socially Relevant Project-I Work

entitled **Voice Enabled ATM**

is the bonafide work done by

G. Naga Eshitha	**19121A0565**
C. Anusha	**19121A0538**
Abdul Bari Shaik	**19121A0502**
G. Hemanth Kumar	**19121A0558**
B. Nagendra Naik	**19121A0524**

In the Department of Computer Science and Engineering, Sree Vidyanikethan Engineering College, A.Rangampet. is affiliated to JNTUA, Anantapuramu in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering during 2021-2022.

This is work has been carried out under my guidance and supervision.

The results embodied in this Socially Relevant Project-I report have not been submitted in any University or Organization for the award of any degree or diploma.

Internal Guide
Dr. K. Reddy Madhavi
Associate Professor
Dept of CSE
Sree Vidyanikethan Engineering College
College Tirupathi

Head
Dr. B. Narendra Kumar Rao
Prof & Head
Dept of CSE
Sree Vidyanikethan Engineering
Tirupathi

INTERNAL EXAMINER

EXTERNAL EXAMINER

References:

This is a list of external material referred to in the Text. The entries should be numbered in the order in which the references occur. These references are indicated in the text by the use of the number either in superscript^3 or bracketed [3] form. This will indicate the complete citation in the reference section. The format of this citation depends on whether it refers to a book or technical paper.

(a) Books: The convention here is:
Eric Matthews, Python crash course (2nd Edition), NoStarch Press, US, 2019, PP 456-512

(b) Books: The convention here is:
Mark Liu, Make python talk, No Starch Press, US, 2021, PP 307-354

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION AND MISSION

VISION

To become a Centre of Excellence in Computer Science and Engineering by imparting high quality education through teaching, training and research.

MISSION

The Department of Computer Science and Engineering is established to provide undergraduate and graduate education in the field of Computer Science and Engineering to students with diverse background in foundations of software and hardware through a broad curriculum and strongly focused on developing advanced knowledge to become future leaders.

Create knowledge of advanced concepts, innovative technologies and develop research aptitude for contributing to the needs of industry and society.

Develop professional and soft skills for improved knowledge and employability of students.

Encourage students to engage in life-long learning to create awareness of the contemporary developments in computer science and engineering to become outstanding professionals.

Develop attitude for ethical and social responsibilities in professional practice at regional, National and International levels.

Program Educational Objectives (PEO’s)

1. Pursuing higher studies in Computer Science and Engineering and related disciplines

2. Employed in reputed Computer and I.T organizations and Government or have established startup companies.

3. Able to demonstrate effective communication, engage in team work, exhibit leadership skills, ethical attitude, and achieve professional advancement through continuing education.

Program Specific Outcomes(PSO's)

1. Demonstrate knowledge in Data structures and Algorithms, Operating Systems, Database Systems, Software Engineering, Programming Languages, Digital systems, Theoretical Computer Science, and Computer Networks. (PO1)

2. Analyze complex engineering problems and identify algorithms for providing solutions (PO2)

3. Provide solutions for complex engineering problems by analysis, interpretation of data, and development of algorithms to meet the desired needs of industry and society. (PO3,PO4)

4. Select and Apply appropriate techniques and tools to complex engineering problems in the domain of computer software and computer based systems (PO5)

Program Outcomes (PO's)

1. Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems (**Engineering knowledge**).

2. Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences (**Problem analysis**).

3. Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations (**Design/development of solutions**).

4. Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions (**Conduct investigations of complex problems**).

5. Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations (**Modern tool usage**)

6. Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice (**The engineer and society**)

7. Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, andneed for sustainable development (**Environment and sustainability**).

8. Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice (**Ethics**).

9. Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings (**Individual and team work**).

10. Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions (**Communication**).

11. Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments (**Project management and finance**).

12. Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change (**Life-long learning**).

Course Outcomes

CO1. Create/Design engineering systems or processes to solve complex societal problems using appropriate tools and techniques following relevant standards, codes, policies, regulations and latest developments. (PO1, PO2, PO3, PO4, PO5, PO6, PO8 & PO12)

CO2. Consider environment, sustainability, economics and project management in addressing societal problems. (PO7 & PO11)

CO3. Perform individually or in a team besides communicating effectively in written, oral and graphical forms on socially relevant project. (PO9 & PO10)

CO-PO Mapping

Course Outcome	Program Outcomes												Program Specific Outcomes			
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
CO1	3	3	3	3	3	3		3				3	3	3	3	3
CO2							3				3		3	3	3	3
CO3									3	3						

(Note: 3-High, 2-Medium, 1-Low)

DECLARATION

We hereby declare that this Socially Relevant Project-I report titled **Voice Enabled ATM** is a genuine Socially relevant project-I work carried out by us, in **B.Tech *(Computer Science and Engineering)*** degree course of **Jawaharlal Nehru Technological University Anantapur** and has not been submitted to any other course or University for the award of any degree by us.

Signature of the student

1.

2.

3.

4.

5.

ACKNOWLEDGEMENT

We are extremely thankful to our beloved Chairman and founder **Dr. M. Mohan Babu** who took keen interest to provide us the infrastructural facilities for carrying out the socially relevant project-I work.

We are highly indebted to **Dr. B. M. Satish**, Principal of Sree Vidyanikethan Engineering College for his valuable support and guidance in all academic matters.

We are very much obliged to **Dr. B. Narendra Kumar Rao,** Professor & Head, Department of CSE, for providing us the guidance and encouragement in completion of this project.

We would like to express our indebtedness to the Project Coordinator, **MS. K. Ghamya**, Assistant Professor, Department of CSE for her valuable guidance during the course of socially relevant project-I work.

We would like to express our deep sense of gratitude to **Dr. K. Reddy Madhavi**, Associate Professor, Department of CSE, for the constant support and invaluable guidance provided for the successful completion of the project.

We are also thankful to all the faculty members of CSE Department, who have cooperated in carrying out our project. We would like to thank our parents and friends who have extended their help and encouragement either directly or indirectly in completion of our socially relevant project-I work.

Abstract

Automated Teller Machine (ATM) has become vital part of our life to perform financial transactions without intervention of human banker. ATM facilitates cash withdrawal, balance check, mini statement and fund transfer. But these banking services using ATM cannot be directly used by some set of people of society such as people with low vision, visually impaired, illiterate as lack of accessing ATM through screens. Even they can be defrauded at ATM centers. To digitally include these set of people, voice enabled ATMs are evolved. Voice enabled ATM provides accessibility to ATM services by providing audio component. Many ATMs employ headphone jack that facilitates user to do transaction with security. The audio information is generated either using pre-recorded speech corpus or through speech synthesis engine. In this project, we are going to interconnect the voice assistant to one of the banking facility which is the ATM machine. First a person will enter into the ATM through door.

The sensor which is attached to the door will detect the person and notifies the voice assistant. Now, the voice assistant will communicate with the person and asks his PIN number. It will recognize the PIN number said by the person from the microphone which is connected to the voice assistant. And later this voice is converted into text and text is given to ATM for checking purpose. ATM will sends the acknowledgement about the PIN as either its true or false. Based on the acknowledgement given by the ATM, voice assistant will act as per the given instructions. And if the PIN given is correct, it will askthe person to tell the amount that he need to draw. It will recognize the audio input or it will show the money on the screen for the confirmation purpose that needed to be drawn. After this confirmation, voice assistant will send the information to the ATM about money that he needs to draw.

ATM will provide the specified money and the person is told to take the money. Finally, it says "Thank you" for us. For this, the proposed solution is to use the ATM using voice assistant.

Table of Contents

List of Tables

List of Figures

1. INTRODUCTION

1.1 Introduction

The proposed project can be considered as ATM and a voice assistant is connected to it. The voice assistant takes the speech as input and coverts to text. Default language is set to it based on the language preferred there. To make this more secure, we can use sensors and allow only one person or we can use OTP instead of ATM pin while using voice. A voice enabled ATM is a type of automated teller machine (ATM) that provides audible instructions so that persons who cannot read an ATM screen can independently use the machine. All audible information is delivered privately through a standard headphone jack on the face of the machine or separately attached telephone handset. Information is delivered to the customer either through pre-recorded sound files or via text-to-speech speech synthesis.

1.2 Problem Statement

Many people use ATM's in order to make transactions but some illiterates doesn't know how to use the ATM and due to covid no one wants to touch the ATM screen. Even visually impaired can't use ATM's as they can't see the screen. By considering all the circumstances, voice assistant can be interlinked with ATM's kernel so that the visually impaired or others can independently make their transactions without others help.

1.3 Objectives

- To help the visually impaired by assisting them with the voice assistant.
- To help the illiterates who can't understand thetransaction process.
- To help the society for performing transactions without touching screen.
- To make the ATM for processing fast transactions.
- To reduce usage of the touch in ATM.
- To make the voice assistant which helps in doing the transactions more secure

1.4 Scope

The present ATM's are based on inserting ATM card and entering PIN for their needs. The user needs to select the options manually. This project is to improvise the ATM into voice enabled ATM in such a way that it recognizes the user option through speech and gives voice instructions to the user to make it easy. It can be further increased into a broader scope by implementing language selection option to make it easier for illiterates and for more convenience.

1.5 Societal applications

- This is the basic model of voice enabled ATM, which can be used for assisting ATM transactions.
- It also provides assistance to visually impaired to make transactions.
- The voice assistant makes the transaction easier and helps to achieve less work-fast transaction.

1.6 Limitations

- The voice assistant should be well trained before deploying.
- Internet connection is needed to recognize the speech commands of the user.
- For security reasons, only one person is allowed to enter to perform transactions.

2. LITERATURE SURVEY

2.1 Introduction

This section summarizes about the existing ATM and the modifications made. This section also summarizes how the newly modeled system can be used and how it improves the interaction with the user.

2.2 Analyzing the Existing System

In the existing system user has to insert the ATM card and they need to enter the PIN, then the card and the PIN entered are validated and if they are correct, the user receives the set of facilities or options and can choose according to his needs.

The improvised ATM assists the user in performing transactions. It can also take the user's speech input which is helpful in present condition due to covid and also for the users who doesn't know how to use it.

3. ANALYSIS

To develop the voice enabled ATM and add the voice assistant, we understood and analyzed the following points.

- Database required to store user details to validate
- Inserting ATM card
- Entering PIN
- Select from options
- Entering required details
- Waiting for ATM to process
- Displaying or performing selected task
- Thankyou message

After understanding the current process in ATM, we analyzed that it can be further improved to help the users and make it easier to use. We can improve the performance and efficiency and user interaction with the system.

Then we analyzed,

- How to add a voice assistant
- Design and model of basic ATM
- What sort of information should be displayed
- How to assist the user
- Storing the information in a database online
- Accepting speech input
- Maintaining privacy and security

4. DESIGN

4.1 Introduction

Detailed design starts after the system design phase and system has been certified through the review. The goal of this phase is to develop the internal logic of each of the modules identified during system design. In the system design, the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for the modules. In other words, in system design attention is on what components are needed, while in the detailed design how the components can be implemented in the software is the issue, the design process for software system has two eves. At the first level focus is on deciding which modules are needed for the system, the specification of these modules and how the modules should be interconnected. This is called system design or top-level design. In the specification of the module can be satisfied is decided. This design level is often called detailed design.

Because the detailed design is extension of system design, system design controls the major structural characteristics of the system. The system design has a major impact testability and modifiability of a system and impacts its efficiency much of the design efforts for the designing software are spent creating the system design.

4.2 Applicable Documentation

The detailed design refers the system document hence the first applicable document here is system design. Also we are referring the data structure. Hence second applicable document here is database design.

4.3 Requirements

4.3.1 Hardware Requirements

- Intel core i5 or higher Processor
- 8 GB RAM or higher
- 100 GB ROM or higher

4.3.2 Software Requirements

- Windows 7 or higher
- Visual Studio
- Sqlite
- Python

4.4 System Design

4.4.1 ER Diagrams

4.4.2 UML Diagrams

4.4.2.1 Use Case Diagram

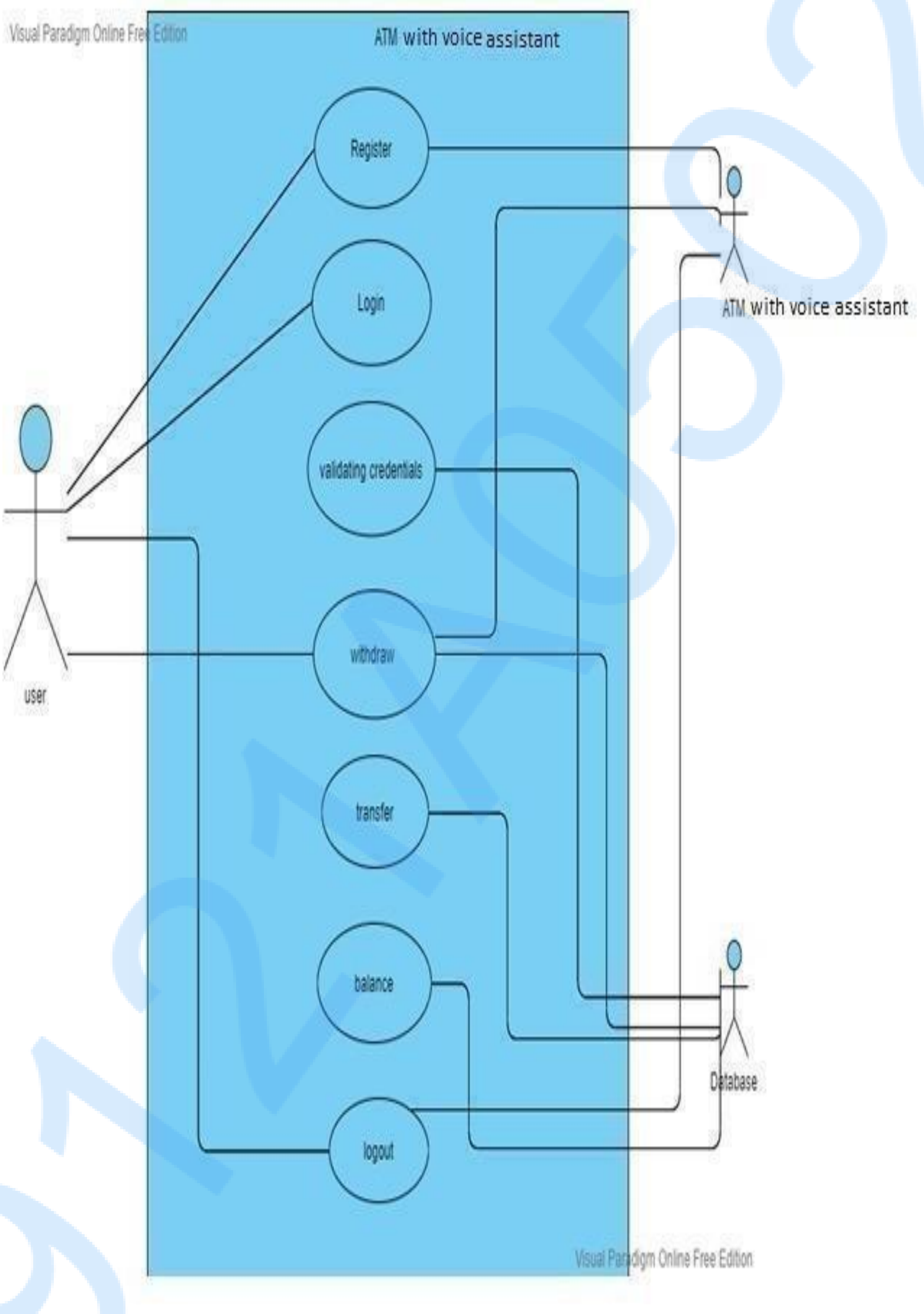


Figure 1: Use Case Diagram

Use case diagrams model the functionality of the system. It consists of actors, use cases and relationships.

Use Cases are:

1. Admin
   - Register
   - Login
   - Validation
   - Withdraw
   - Transfer
   - Balance
   - Logout

2. Actors
    - User
    - ATM with voice assistant
    - Database
3. Relationships
    - Association

4.4.2.2 Class Diagram

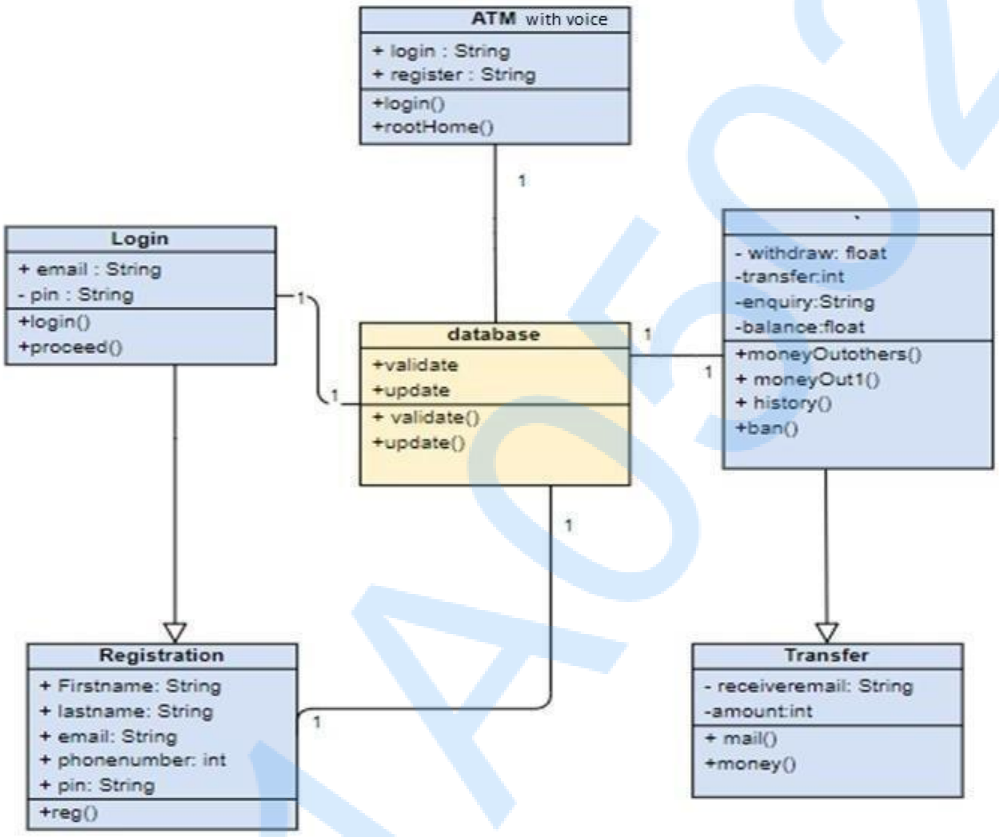
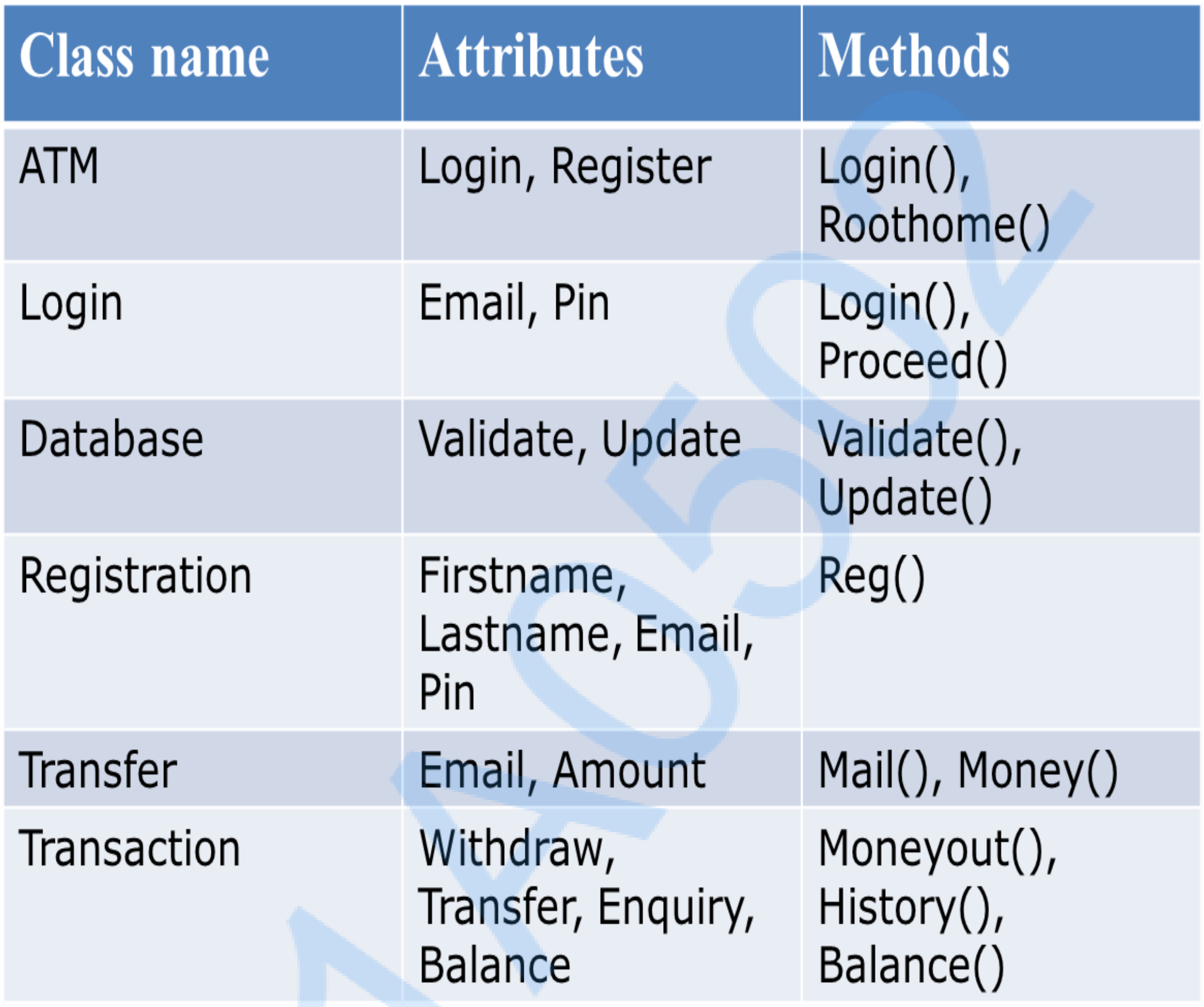


Figure 2: Class Diagram

Class diagram describes the structure of a system by showing the system classes, attributes, operations and relationships among them Relationships are:

- Association
- Dependency



Class name	Attributes	Methods
ATM	Login, Register	Login(), Roothome()
Login	Email, Pin	Login(), Proceed()
Database	Validate, Update	Validate(), Update()
Registration	Firstname, Lastname, Email, Pin	Reg()
Transfer	Email, Amount	Mail(), Money()
Transaction	Withdraw, Transfer, Enquiry, Balance	Moneyout(), History(), Balance()

Figure 3: Use Cases - Attributes and Methods

4.4.2.3 Sequence Diagram

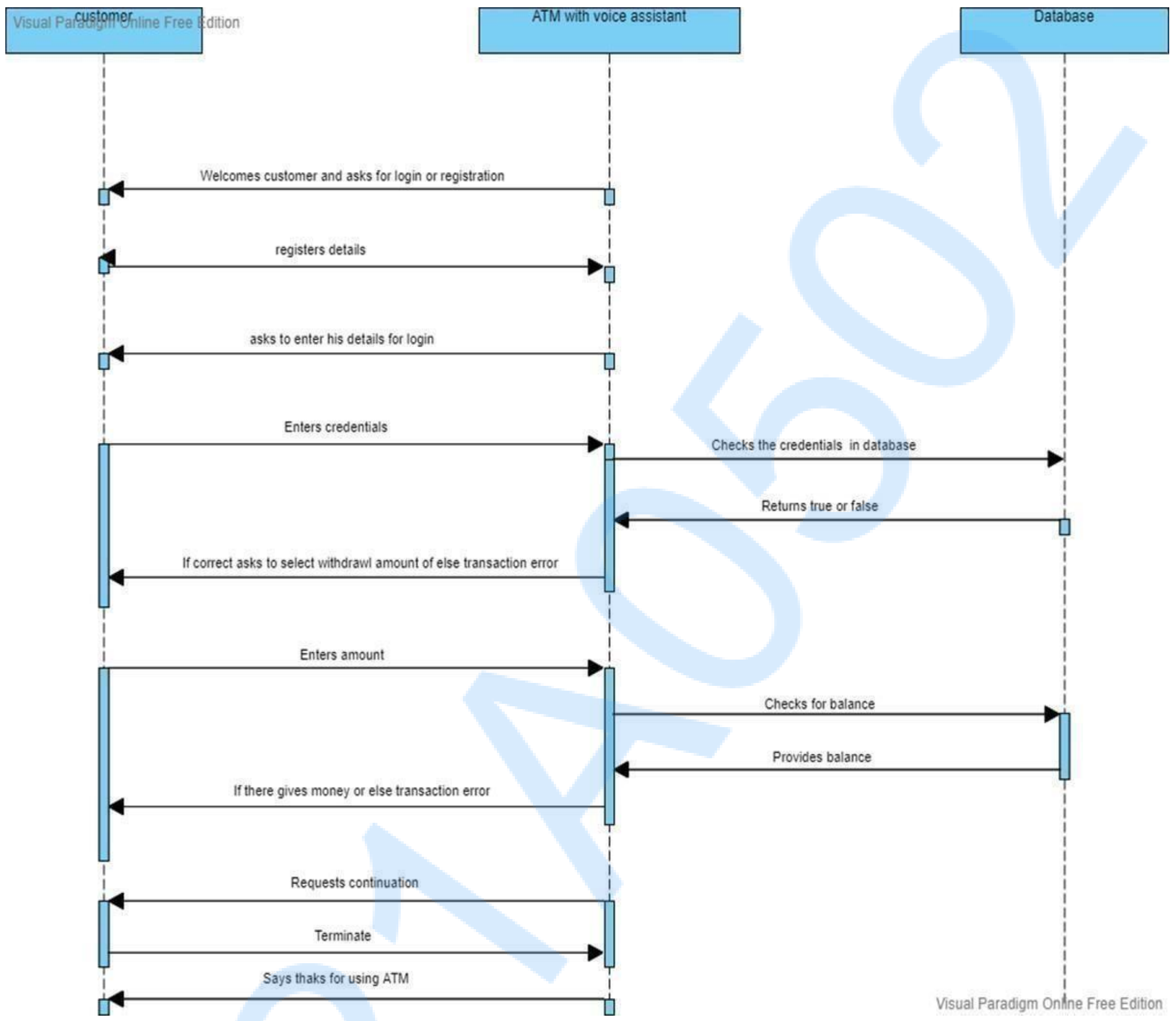

customer
Visual Paradigm Online Free Edition
ATM with voice assistant
Database
Welcomes customer and asks for login or registration
registers details
asks to enter his details for login
Enters credentials
Checks the credentials in database
Returns true or false
If correct asks to select withdrawl amount of else transaction error
Enters amount
Checks for balance
Provides balance
If there gives money or else transaction error
Requests continuation
Terminate
Says thaks for using ATM
Visual Paradigm Online Free Edition

Figure 4: Sequence Diagram

It depicts the interaction between objects in a sequence order.

1. Objects
   - User
   - ATM with voice assistant
   - Database
2. Messages
   - Welcomes customer and asks for login or register
   - Register details
   - Check credentials in database
   - If correct asks for the options to select

- Performs action
- Checks balance
- Provides balance
- Requests continuation
- Terminate
- Voice message saying thankyou

4.4.2.4 Activity Diagram

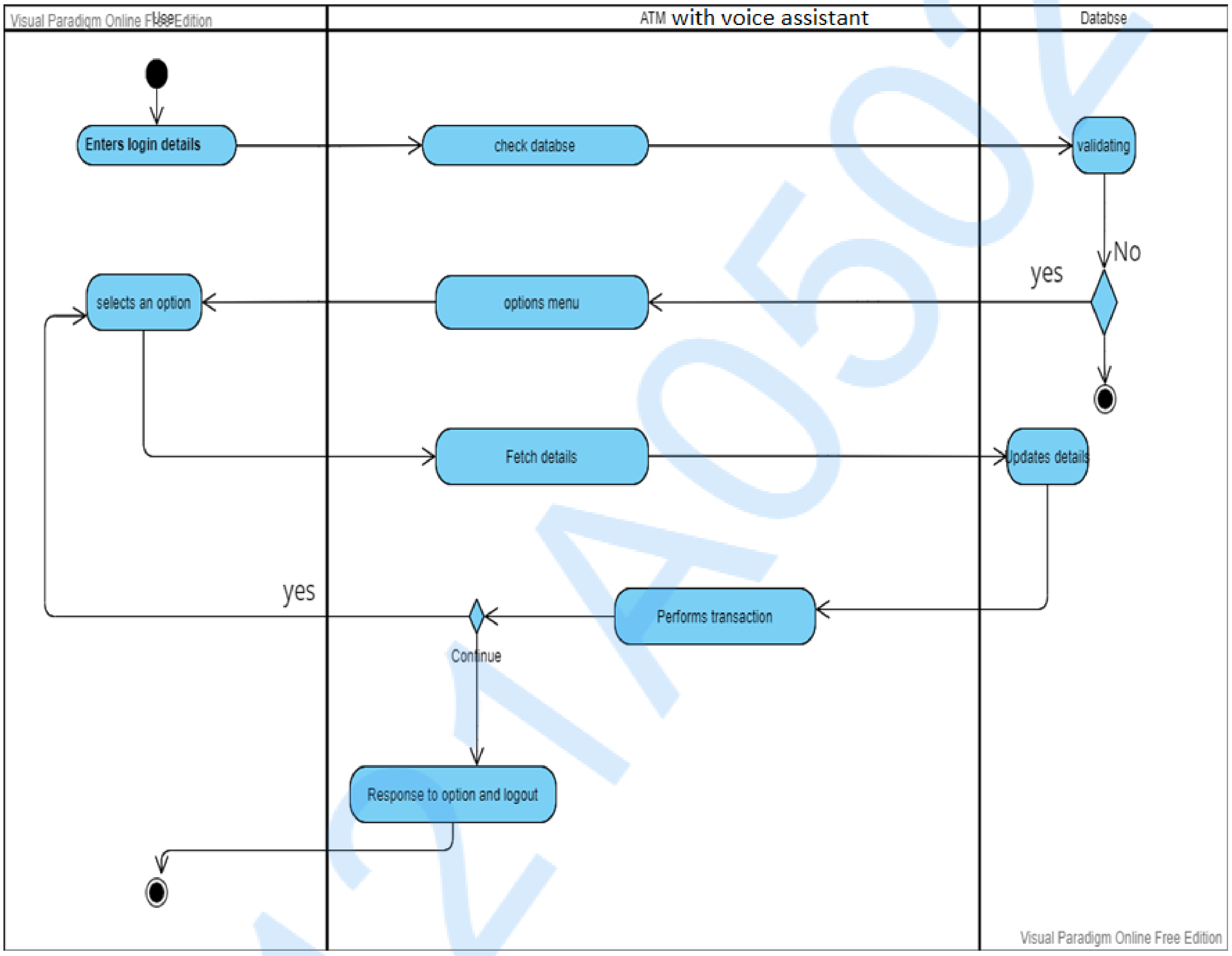


Figure 5: Activity Diagram

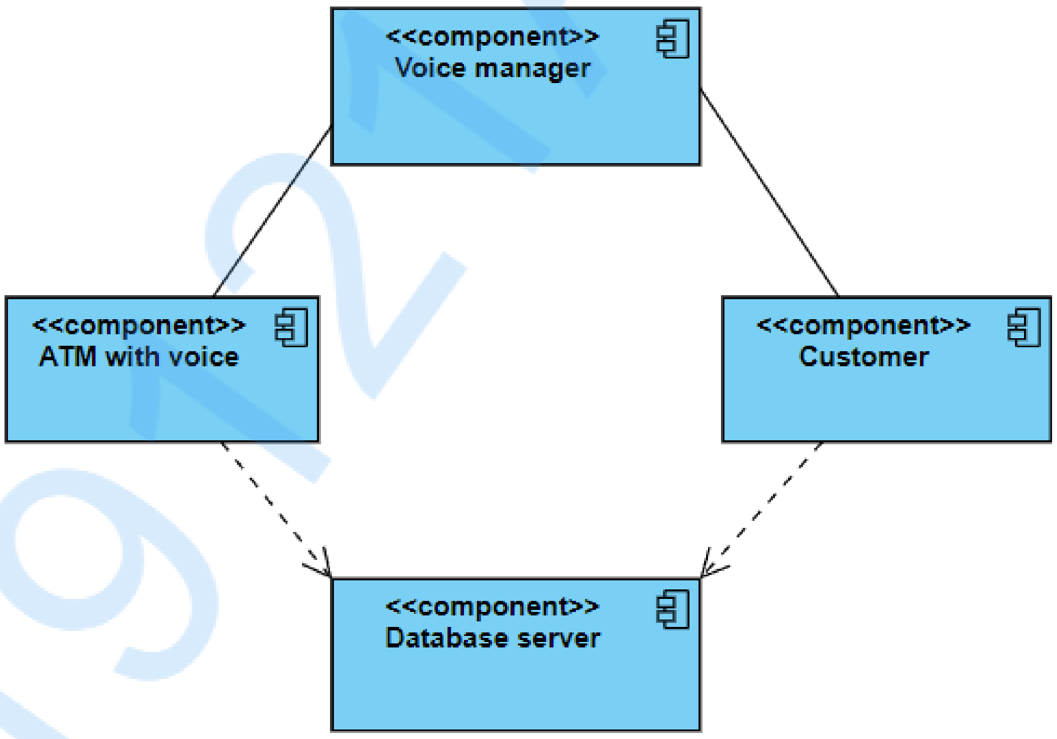
It depicts flow from one activity to another.

Activities are:

1. User
    - Login
    - Select option
    - Logout

2. ATM
    - Validating by checking database
    - Display options menu
    - Fetch details
    - Perform transaction
    - Logout
3. Student
    - Store details
    - Validate
    - Update details

4.4.2.5 Component Diagram



<<component>> Voice manager
<<component>> ATM with voice
<<component>> Customer
<<component>> Database server

Figure 6: Component Diagram

These are used to visualize, organizations and relationships among components.

1. Components
    - Event Management System
    - Admin
    - Student
    - Database Server
2. Relationships
    - Association
    - Dependency

4.4.2.6 Deployment Diagram

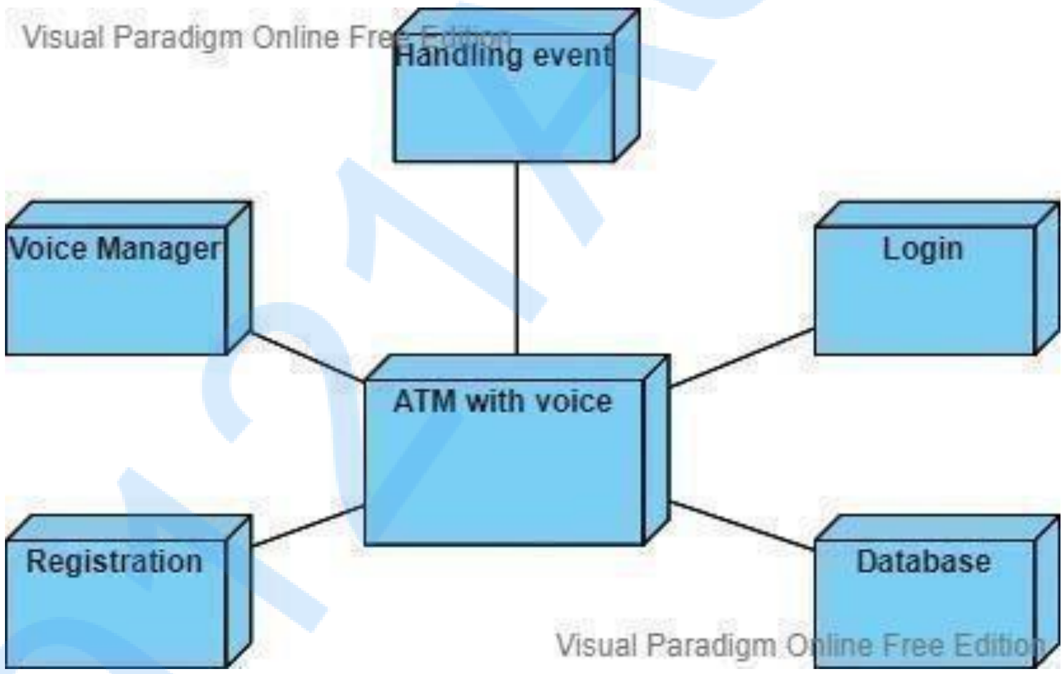

Visual Paradigm Online Free Edition
Handling event
Voice Manager
Login
ATM with voice
Registration
Database
Visual Paradigm Online Free Edition

Figure 7: Deployment Diagram

It represents the deployment view of a system. It consists of nodes which are used to deploy the application.

1. Nodes
    - ATM with voice
    - Registration
    - Login
    - Event Handling
    - Database
    - Voice Manager
2. Relationships
    - Association

4.4.3 Execution Steps Step

1: Creating the Database Table For

storing details of users

Example:

```
CREATE TABLE users (email INT NOT NULL PRIMARY KEY
UNIQUE, username VARCHAR(50) NOT NULL UNIQUE, password
VARCHAR(255) NOT NULL);
```

Step 2: Creating the

Connection with database

For Example:

```
$con = mysqli_connect('localhost','root','');

mysqli_select_db($con,'dbfs');
```

Step 3: Voice assistance using pyttsx3

For Example:

```
import pyttsx3
speak(“Welcome”)
```

Step 4: Create a code for validating entered details with the details in database

For example:

```
val=conn.execute("SELECT Balance FROM datab WHERE
Emailid=?",(email,)).fetchone()
```

4.5 Database Design

Data Design is used to manage large amount of information. The management of data involves both the definition of structure for storage and provision for the manipulation of information.

4.5.1 Table Structure

Database is a collection of information and data systematically stored in tables . Table is a list of information organized into fields. Usually each field has a field name, data type with fixed length and description.

- Field name tells the name of the field
- Data type's property tells the type of data stored in that field and the length is specified, so that the maximum length is set
- Attribute property is mentioned so those mandatory fields are mentioned
- Description field tells the detail of field

Table 1: User Details

Fields	**Data type**	**Constraints**	**Description**
Emailid	Varchar(20)	Primary key	To store the ID of the user.
First Name	Varchar(20)	Not Null	To store user's first name
Last Name	Varchar(20)	Not Null	To store user's last name
Username	Varchar(20)	Not Null	To store username
PIN	INT	Not Null	To store PIN of the user
Balance	INT	Not Null	To store balance of the user

5. CODING

5.1 Introduction

The goal of coding or programming phase is to translate the design of the system produced during the design phase into code in a given programming language, which can be executed by a computer and that performs the computation specified by the design. The coding phase is not to simplify the job of the programmer. Rather, the goal should be simplifying the job of the tester and maintainer.

There are many different criteria for judging a program, including readability, size of program, execution time and required memory. Having readability and understanding ass a clear objective of the coding activity can itself help in producing software that is more maintainability.

The coding is done with the following characteristics in mind:

- Ease of design to code translation
- Code efficiency
- Memory efficiency
- Response time
- Maintainability
- Security
- Simple ease to understand code
- Efficient and consistentlogic Programming style

It is impossible to provide an exhaustive list of what to do and not to do produce simple readable

Code: Next we will some general rules that usually apply.

Name

Selecting module and variable names is often not considered important by invoice programmers. Most variable in a program reflect some entity in the program domain, and the module reflect some process. Variable names should be closely related the entity they represent and module name should be reflected their activity.

Control constructs

It is desirable to use a few standard control constructs rather than using a wide variety of constructs, just because their available in the language.

USER DEFINED TYPES

Modern languages allow user to defined types like the enumerated type. When such facilities are available, they should be exploited where applicable.

Module size

A programmer should examine any routine with very few statements or too many statements. Large modules often will not be functionally cohesive, and too small modules might incur unnecessary overhead.

Module interface

A module with a complex interface should be carefully examined. Such modules might not be functionally cohesive and might be implementing multiple functions.

Robustness

A program is robust if it does something planned for exceptional conditional. A program might encounter exceptional condition in such from as incorrect input, the correct value of some variable and overflow. A program should try to handle such situation. In general, a program should check for validation, where possible and should check for possible, and should check for possible overflow of data structures.

5.2 Codes

```
import speech_recognition as sr
import pyttsx3
import sqlite3
import my_module

generalList = []

data = {}

email = ""

pin  = ""

amountValue = 0

cardOption = ""

engine=pyttsx3.init()

voices=engine.getProperty('voices')

engine.setProperty('voice','voices[0].id')

conn = sqlite3.connect('datab.db')

cursor = conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS datab

            (Fname char(20), Lname char(20), Uname char(20),
Emailid char(20), Pin n(5), Balance n(5));''')


def speak(text):

    engine.say(text)

    engine.runAndWait()

def capture():

    """Capture audio"""
    rec = sr.Recognizer()

    with sr.Microphone() as source:
        rec.adjust_for_ambient_noise(source, duration=0.5)
        rec.dynamic_energy_threshold = True
        audio = rec.listen(source,phrase_time_limit=10)

    try:
        text = rec.recognize_google(audio,language='en-in')
        return text

    except:
        text=input()
        return text

def generallist():
    generalOption = capture()
    if generalOption == 3:
        print(generalList)
        rootHome()
    else:
        print("error")
        speak('error')

def logOut():
    speak('Thank You for banking with us, have a nice day')
    print("Thank You for banking with us, have a nice day.")
    conn = sqlite3.connect('datab.db')

    cursor = conn.cursor()

    cursor.execute('''CREATE TABLE IF NOT EXISTS datab

            (Fname char(20), Lname char(20), Uname char(20),
Emailid char(20), Pin n(5), Balance n(5));''')


    speak('Welcome enter 1 for login enter 2 for register')

    print("Welcome to ")

    print("1. Login")

    print("2. Register")


    rootHome()


def moneyOut1():

    global email

    global pin

    message = ""
    a=conn.execute("SELECT Balance FROM datab WHERE
Emailid=?",(email,)).fetchone()
    b=a[0]


    if int(b) < 10000:

        print("Insufficient Balance")

        speak('Insufficient balance')

    elif int(b) > 10000:

        val=int(b)  - 10000

        print("Your #10000 withdrawal was successful.")

        speak('Your 10000 withdrawal was successful')
        conn.execute("UPDATE datab SET Balance=? WHERE
Emailid=?",(val,email))
        conn.commit()

        print("Your current balance is #" + str(val) + ".")

        speak('Your current balance is '+str(val))

    else:

        message = "Transaction error"

        speak(message)



def moneyOut2():

    global email

    global pin

    message = ""
    a=conn.execute("SELECT Balance FROM datab WHERE
Emailid=?",(email,)).fetchone()
    b=a[0]

    if int(b) < 5000:

        print("Insufficient Balance")

        speak('Insufficient balance')

    elif int(b) > 5000:

        val=int(b) - 5000

        print("Your #5000 withdrawal was successful.")

        speak("Your #5000 withdrawal was successful.")

        conn.execute("UPDATE datab SET Balance=? WHERE
Emailid=?",(val,email))
        conn.commit()

        print("Your current balance is #" + str(val) + ".")

        speak('Your current balance is '+str(val))

    else:

        message = "Transaction error"
        speak(message)


def moneyOut3():

    global email

    global pin

    message = ""
    a=conn.execute("SELECT Balance FROM datab WHERE
Emailid=?",(email,)).fetchone()
    b=a[0]

    if int(b) < 2000:

        print("Insufficient Balance")

        speak('Insufficient balance')

    elif int(b) > 2000:

        val=int(b) - 2000

        print("Your #2000 withdrawal was successful.")

        speak('Your #2000 withdrawal was successful.')

        conn.execute("UPDATE datab SET Balance=? WHERE
Emailid=?",(val,email))
        conn.commit()

        print("Your current balance is #" + str(val) + ".")

        speak('Your current balance is '+str(val))

    else:

        message = "Transaction error"

        speak(message)



def moneyOut4():

    global email
    global pin

    message = ""
    a=conn.execute("SELECT Balance FROM datab WHERE
Emailid=?",(email,)).fetchone()
    b=a[0]

    if int(b) < 1000:

        print("Insufficient Balance")

        speak('Insufficient balance')

    elif int(b) > 1000:

        val=int(b) - 1000

        print("Your #1000 withdrawal was successful.")

        speak('Your #1000 withdrawal was successful.')

        conn.execute("UPDATE datab SET Balance=? WHERE
Emailid=?",(val,email))
        conn.commit()

        print("Your current balance is #" + str(val) + ".")

        speak('Your current balance is '+str(val))

    else:

        message = "Transaction error"

        speak(message)


def moneyOut5():

    global email

    global pin
    message = ""
    a=conn.execute("SELECT Balance FROM datab WHERE
Emailid=?",(email,)).fetchone()
    b=a[0]



    if int(b) < 500:

        print("Insufficient Balance")
        speak('Insufficient balance')

    elif int(b) > 500:

        val=int(b) - 500

        print("Your #500 withdrawal was successful.")

        speak('Your #500 withdrawal was successful.')

        conn.execute("UPDATE datab SET Balance=? WHERE
Emailid=?",(val,email))
        conn.commit()

        print("Your current balance is #" + str(val) + ".")

        speak('Your current balance is '+str(val))

    else:

        message = "Transaction error"

        speak(message)




def moneyOutothers():

    global email

    global pin

    global amountValue

    message = ""
    a=conn.execute("SELECT Balance FROM datab WHERE
Emailid=?",(email,)).fetchone()
    b=a[0]



    if b < amountValue:

        print("Insufficient Balance")

        speak('Insufficient balance')

    elif b > amountValue:
        val=int(b) - amountValue

        print("Your #" + str(amountValue) + " withdrawal was
successful.")

        speak('Your'+str(amountValue)+ 'withdrawal was
successful')

        conn.execute("UPDATE datab SET Balance=? WHERE
Emailid=?",(val,email))
        conn.commit()

        print("Your current balance is #" + str(val) + ".")

        speak('Your current balance is '+str(val))

    else:

        message = "Transaction error"

        speak(message)


def withdraw():

    speak('enter how much you want to withdraw')

    print("How much do you want to withdraw?")

    print("1. 10,000")

    speak('enter 1 for 10000')

    print("2. 5,000")

    speak('enter 2 for 5000')

    print("3. 2,000")

    speak('enter 3 for 2000')

    print("4. 1,000")

    speak('enter 4 for 1000')

    print("5. 500")

    speak('enter 5 for 500')

    print("6. others")
    speak('enter 6 for others')

    print("7. Back")

    speak('enter 7 to go back')

    print("Select option: ")

    withdrawOption = capture()

    if withdrawOption == "1" or withdrawOption == 1:


        moneyOut1()

        print("Do you want to perform another transaction?")

        speak('Do you want to perform another transaction?')

        print("1. Yes")

        speak('enter 1 if yes')

        print("2.  No")

        speak('enter 2 if no')

        print("Select option: ")

        anotherOption = capture()

        if anotherOption == "1" or anotherOption ==1:

            user()

        elif anotherOption == "2" or anotherOption =="Tu" or
anotherOption ==2:

            logOut()
            conn.commit()
            cursor.close()
            conn.close()

        else:

            print("Invalid selection")

            speak('Invalid selection')

    elif withdrawOption == "2" or withdrawOption =="Tu" or
withdrawOption ==2:
        moneyOut2()

        print("Do you want to perform another transaction?")

        speak('Do you want to perform another transaction?')

        print("1. Yes")

        speak('enter 1 if yes')

        print("2.  No")

        speak('enter 2 if no')

        print("Select option: ")

        anotherOption = capture()

        if anotherOption == "1" or anotherOption ==1:

            user()

        elif anotherOption == "2" or anotherOption =="Tu" or
anotherOption ==2:

            logOut()
            conn.commit()
            cursor.close()
            conn.close()

        else:

            print("Invalid selection")

            speak('Invalid selection')

    elif withdrawOption == "3" or withdrawOption =="tree" or
withdrawOption ==3:


        moneyOut3()

        print("Do you want to perform another transaction?")

        speak('Do you want to perform another transaction?')

        print("1. Yes")
        speak('enter 1 if yes')

        print("2.  No")

        speak('enter 2 if no')

        print("Select option: ")

        anotherOption = capture()

        if anotherOption == "1" or anotherOption ==1:

            user()

         elif anotherOption == "2" or anotherOption ==2:

            logOut()
            conn.commit()
            cursor.close()
            conn.close()

        else:

            print("Invalid selection")

            speak('Invalid selection')

    elif withdrawOption == "4" or withdrawOption ==4:


        moneyOut4()

        print("Do you want to perform another transaction?")

        speak('Do you want to perform another transaction?')

        print("1. Yes")

        speak('enter 1 if yes')

        print("2.  No")

        speak('enter 2 if no')

        print("Select option: ")

        anotherOption = capture()

        if anotherOption == "1" or anotherOption ==1:

            user()
```

```
  elif anotherOption == "2" or anotherOption ==2:

      logOut()
      conn.commit()
      cursor.close()
      conn.close()

  else:

      print("Invalid selection")

      speak('Invalid selection')

elif withdrawOption == "5" or withdrawOption ==5:


  moneyOut5()

  print("Do you want to perform another transaction?")

  speak('Do you want to perform another transaction?')

  print("1. Yes")

  speak('enter 1 if yes')

  print("2.  No")

  speak('enter 2 if no')

  print("Select option: ")

  anotherOption = capture()

  if anotherOption == "1" or anotherOption ==1:

      user()

   elif anotherOption == "2" or anotherOption ==2:

      logOut()
      conn.commit()
      cursor.close()
      conn.close()

  else:

      print("Invalid selection")

      speak('Invalid selection')
```

```
elif withdrawOption == "6" or withdrawOption ==6:

    global amountValue

    amountValue = int(input())

    moneyOutothers()

    print("Do you want to perform another transaction?")

    speak('Do you want to perform another transaction?')

    print("1. Yes")

    speak('enter 1 if yes')

    print("2.  No")

    speak('enter 2 if no')

    print("Select option: ")

    anotherOption = capture()

    if anotherOption == "1" or anotherOption ==1:

        user()

    elif anotherOption == "2" or anotherOption ==2:

        logOut()
        conn.commit()
        cursor.close()
        conn.close()

    else:

        print("Invalid selection")

        speak('Invalid selection')

elif withdrawOption == "7" or withdrawOption ==7:

    user()

else:

    print("Invalid selection")

    speak("Invalid selection")
```

```
def histroy():

    global email

    global pin

    message = " "

    message = "Still working on it"

    print(message)

    speak(message)


    speak('Do you want to perform another transaction?')

    print("Do you want to perform another transaction?")

    speak('Enter 1 for yes')

    print("1. Yes")

    speak('enter 2 for no')

    print("2.  No")

    print("Select option: ")

    anotherOption = capture()

    if anotherOption == "1" or anotherOption ==1:

        user()

    elif anotherOption == "2" or anotherOption ==2:

        logOut()
        conn.commit()
        cursor.close()
        conn.close()

    else:

        print("Invalid selection")

        speak('Invalid selection')
```

```
def balance():

    global email

    global pin

    message = ""
    val=conn.execute("SELECT Balance FROM datab WHERE
Emailid=?",(email,)).fetchone()
    print("Your current balance is #" + str(val) + ".")

    speak('Your current balance is '+str(val))



    speak('Do you want to perform another transaction?')

    print("Do you want to perform another transaction?")

    speak('Enter 1 for yes')

    print("1. Yes")

    speak('enter 2 for no')

    print("2.  No")

    print("Select option: ")

    anotherOption = capture()

    if anotherOption == "1" or anotherOption ==1:

        user()

    elif anotherOption == "2" or anotherOption =="Tu" or
anotherOption ==2:

        logOut()
        conn.commit()
        cursor.close()
        conn.close()

    else:

        print("Invalid selection")

        speak('Invalid selection')
```

```
def proceed():

    global email
    global pin

     #word = ""

    speak('enter receivers mail address')

    print("Enter the receiver's email address: ")

    remail = capture()

    speak('enter amount you want to transfer')

    print("Enter the amount you want to transfer: ")

    amount = capture()
    a=conn.execute("SELECT Balance FROM datab WHERE
Emailid=?",(email,)).fetchone()
    b=a[0]
    c=conn.execute("SELECT Balance FROM datab WHERE
Emailid=?",(remail,)).fetchone()
    d=c[0]
    f=conn.execute("SELECT FNAME FROM datab WHERE
Emailid=?",(remail,)).fetchone()
    l=conn.execute("SELECT LNAME FROM datab WHERE
Emailid=?",(remail,)).fetchone()
    if int(b)>0 and int(b)>int(amount):
        val=int(b)-int(amount)
        val2=int(d)+int(amount)
        conn.execute("UPDATE datab SET Balance=? WHERE
Emailid=?",(val,email))
        conn.commit()
        conn.execute("UPDATE datab SET Balance=? WHERE
Emailid=?",(val2,remail))
        conn.commit()
        f=f[0]
        l=l[0]

        mess = "#" + amount + " transfered successfully to " + f +
" " + l + "."

    else:

        speak('Insufficient balance')

        print("Insufficient  balance")

        # else:

        #     word = "Insufficient fund"
    # for details in generalList:

    #    if(details["Email"] == email):

    #        details["Balance"] = details["Balance"] + int(amount)

    #        print("#" + amount + " was transfered successfully to "
+ email + ".")

    #    else:

    #        message="Invalid account details"

    # for details in generalList:

    #     if details["Pin"] == pin:

    #         details["Balance"] = details["Balance"] - int(amount)

    #         if details["Balance"] < 0:

    #             print("Insufficient fund")

    #         else:

    #             print("Your current balance is #" +
str(details["Balance"]) + ".")


    speak('Do you want to perform another transaction?')

    print("Do you want to perform another transaction?")

    speak('Enter 1 for yes')

    print("1. Yes")

    speak('enter 2 for no')

    print("2.  No")

    print("Select option: ")

    anotherOption = capture()

    if anotherOption == "1" or anotherOption ==1:

        user()
    elif anotherOption == "2" or anotherOption ==2:

        logOut()
        conn.commit()
        cursor.close()
        conn.close()

    else:

        print("Invalid selection")

        speak('Invalid selection')


def trans():

    speak('enter 1 to proceed enter 2 to go back')

    print("1. Proceed")

    print("2. Back")

    print("Select option: ")

    transOption = capture()

    if transOption == "1" or transOption==1:

        proceed()

    elif transOption == "2" or transOption ==2:

        user()

    else:

        print("Invalid selection")

        speak('Invalid selection')




def user():

    global email
    a=conn.execute("SELECT Fname FROM datab where 
Emailid=?",(email,)).fetchone()
    f=a[0]
    b=conn.execute("SELECT Lname FROM datab where
Emailid=?",(email,)).fetchone()
    l=b[0]

    print("Welcome " + f + l + ".")

    speak('Welcome ' + f + l )

    speak('enter 1 for withdraw')

    print("1. Withdraw")

    speak('enter 2 for balance')

    print("2. Balance")

    speak('enter 3 for Transfer')

    print("3. Transfer")

    speak('enter 4 for balance statement')

    print("4. Histroy")

    speak('enter 5 for logout')

    print("5. Log Out")


    print("Select option: ")

    userOption = capture()


    if userOption == "1" or userOption ==1:

        withdraw()

    elif userOption == "2" or userOption ==2:

        balance()
    elif userOption == "3" or userOption ==3:

        trans()
    elif userOption == "4" or userOption ==4:

        histroy()

    elif userOption == "5" or userOption ==5:
        logOut()
        conn.commit()
        cursor.close()
        conn.close()

    else:

        speak('Invalid selection')

        print("Invalid selection")



def login():

    global email

    global pin

    message = ""

    speak('enter your email')

    print("Enter your Email: ")

    email = capture()

    speak('enter your pin')

    print("Enter your Pin: ")

    pin = capture()
    var=cursor.execute("SELECT Pin FROM datab WHERE
Emailid=?",(email,),).fetchone()
    var1=var[0]
    print(var1)
    if(int(var1)==int(pin)):
        user()
    else:

        message = "Invalid login details"

        print(message)

        speak(message)
```

```
# def access():

#     if generalList[0]["Email"] == "email" and
generalList[0]["Pin"] == "pin":

#         user()

#     else:

#         print("Invalid login details")


def reg():

    global data

    data = {}

    print("Enter first name:")

    speak('Enter first name')

    data["First Name"] = capture()

    print(data["First Name"])

    print("Enter Last Name:")

    speak('Enter last name')

    data["Last Name"] = capture()

    print(data["Last Name"])

    print("Enter your user name: ")

    speak('Enter user name')

    data["User Name"] = capture()

    print(data["User Name"])

    print("Enter your email address: ")

    speak('Enter email')

    data["Email"] = capture()
    print(data["Email"])

    print("Enter your pin:")

    speak('Enter pin')

    data["Pin"] = capture()

    print(data["Pin"])

    print("Enter your balance:")

    speak('Enter balance')

    data["Balance"] = int(capture())

    print(data["Balance"])

    data["History"] = []

    cursor.execute("""

    INSERT INTO datab

    (Fname,Lname,Uname,Emailid,Pin,Balance)

    VALUES (?,?,?,?,?,?)

    """, (data["First Name"],data["Last Name"],data["User
Name"],data["Email"],data["Pin"],data["Balance"]))


    conn.commit()

    var1="User Name" in data

    cursor1=conn.execute("SELECT * FROM datab ORDER BY
rowid DESC limit 1")

    for i in cursor1:

        print(i)



    # for details in generalList:

    #     if(details["Email"] == data["Email"]):

    #         print("Used email address, input a new one.")

    #     else:
```

```
    generalList.append(data)

    speak('welcome to your first transaction')

    print("Welcome "+ data["First Name"] + " " + data["Last
Name"] +".")

    speak('if you wish to transact enter 1 otherwise enter 2')

    print("Do you wish to transact? ")

    print("1. Yes")

    print("2. No")

    print("Select option: ")

    regOption = capture()

    if regOption == "1" or regOption ==1:

        login()

    elif regOption == "2" or regOption ==2:

        logOut()
        conn.commit()
        cursor.close()
        conn.close()

    else:

        print("Invalid selection")

        speak('Invalid selection')


def rootHome():

    homeOption =my_module.capture()

    if homeOption == "1" or homeOption ==1:

        login()

    elif homeOption ==2 or homeOption =="2":
        reg()

    elif homeOption ==3 or homeOption =="3":

        generallist()

    else:

        print("Invalid selection")

        speak('Invalid selection')





print("Welcome to ")

speak("welcome")

speak("enter 1 for login enter 2 for register")

print("1. Login")

print("2. Register")
cursor1=conn.execute("SELECT * FROM datab")
for i in cursor1:
    print(i)

rootHome()
```

6. Testing

6.1 Introduction

Testing is a process used to help identify the correctness, completeness and quality of developed computer software. With that in mind establish the correctness of computer software.

6.2 Objectives

Testing objectives include:

a. Testing is a process of executing a program with the intent of finding an error
b. A good test case is the one that uncovers as yet undiscovered error
c. A successful test is the one that uncovers as yet undiscovered errors

Testing should systematically uncover different classes of errors in a minimum amount time and with a minimum amount of effort. A secondary benefit testing is that it demonstrates that the software appears to be working as stated in the specification.

6.3 Test Methodology

Testing can be done in any one of the following levels:

- Unit testing test the minimal software component or module. Each unit (basic component) of the software is tested to verify that the detailed design for the unit has been correctly implemented.
- Integration testing exposes defects in the interfaces and interfaces and interaction between the integrated components (modules).
- System testing tests a completely integrated system to verify that in meets its requirements.
- System integration testing verifies that a system is integrated

to any external or third party system defined in the system requirements.

Table 2: Test Cases for Login

SERIAL NO	TEST CONDITION	EXPECTED RESULT	TEST RESULT
1	When the user enters valid username and password.	Logs on to the ATM with voice assistant with welcome message	Successful
2	When the user enters " invalid username and password.	ATM with voice assistant says Invalid credentials and displays error message	Successful
3	When the user enters no details	ATM with voice assistant waits until input is given	Successful

Table 3: Test Cases for Register

SERIAL NO	TEST CONDITION	EXPECTED RESULT	TEST RESULT
1	When the user selects "register" option with a valid details.	Logs on to the ATM with voice assistant	Successful
2	When the user selects "register" option with invalid details.	Asks to enter the fields correctly, doesn't redirect to next options	Successful
3	When the user interrupts with a key while entering	Details are not stored in database	Successful

Table 4: Test Cases for Database

SERIAL NO	TEST CONDITION	EXPECTED RESULT	TEST RESULT
1	When the user enters the details by registering.	A new row is added and stores the details in database.	Successful
2	When the user withdraws money	Balance is updated in database and displays the remaining balance.	Successful
3	When user transfers money by having sufficient balance and no sufficient balance	Both account balances are updated. If the balance is not sufficient then says "Insufficient balance"	Successful

Table 5: Test Cases for Voice assistant

SERIAL NO	TEST CONDITION	EXPECTED RESULT	TEST RESULT
1	When user logins	Says "Welcome user name"	Successful
2	When user logouts	Says "Thank you for banking"	Successful
3	When user selects options using speech	Recognizes correctly	Successful
4	When user performs any function	Says the outcome after processing	Successful

7. Conclusion

7.1 Conclusion

We have achieved the drawback of recognizing voice and assisting the users using voice. It is achieved using speech recognition moduleand pyttsx3 packages in python.An ATM model is built using the python in Visual studio codewhich can perform all the functions performed by the ATM.It takes both speech input or text input given by the user .
SQL database is used to store the data of the users in order to validate the login details of the user.

7.2 Future Enhancements

The future enhancements for the ATM with voice assistant can be:

Extending the authentication with iris or fingerprint so there willbe no need of card.

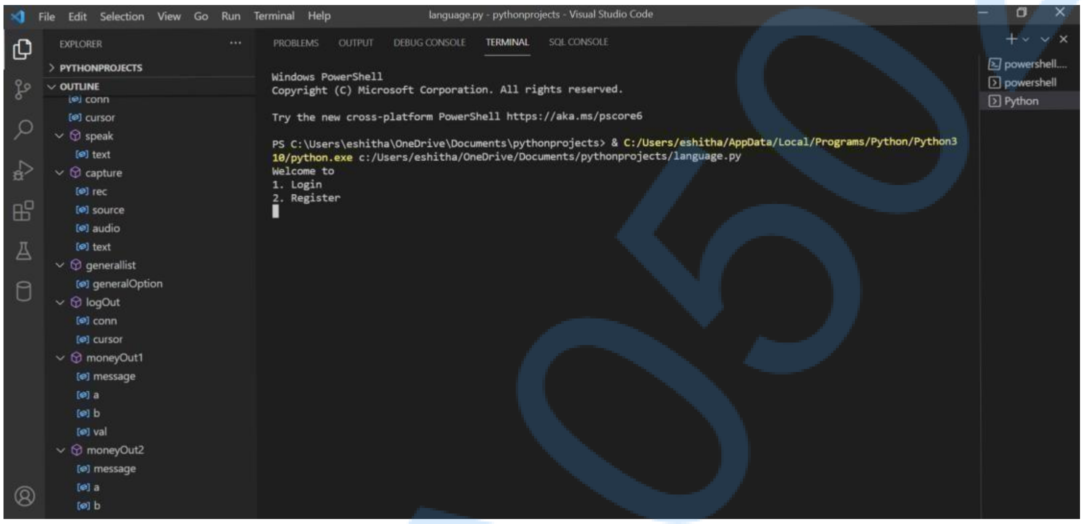
Automatic registration by using only phone number

OTP can be used instead of PINs

To have voice assistant to recognize multiple languages

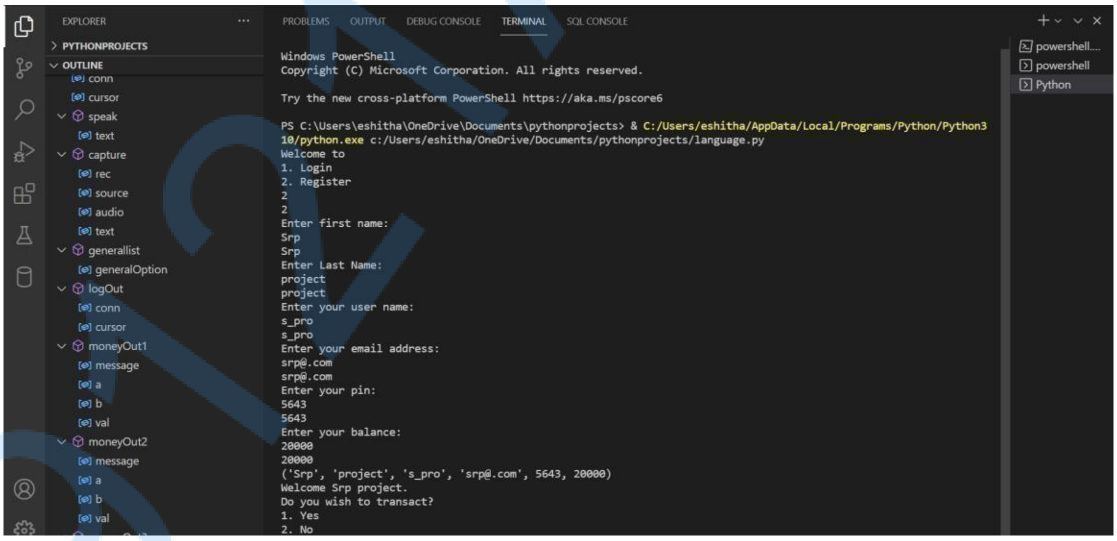
8. Appendix

8.1 Screenshots



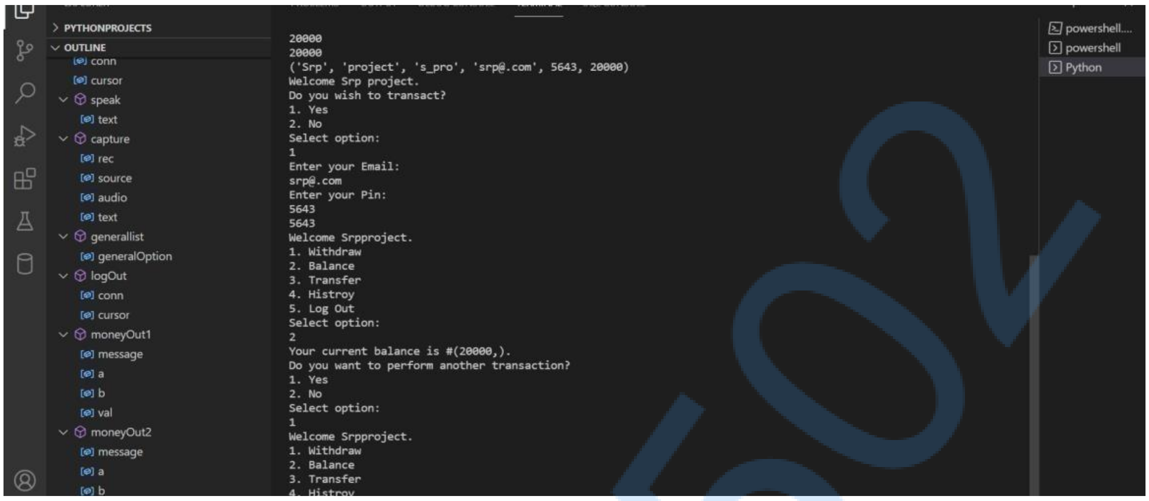
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\eshitha\OneDrive\Documents\pythonprojects> & C:/Users/eshitha/AppData/Local/Programs/Python/Python3
10/python.exe c:/Users/eshitha/OneDrive/Documents/pythonprojects/language.py
Welcome to
1. Login
2. Register
```

Figure 8: Home Interface



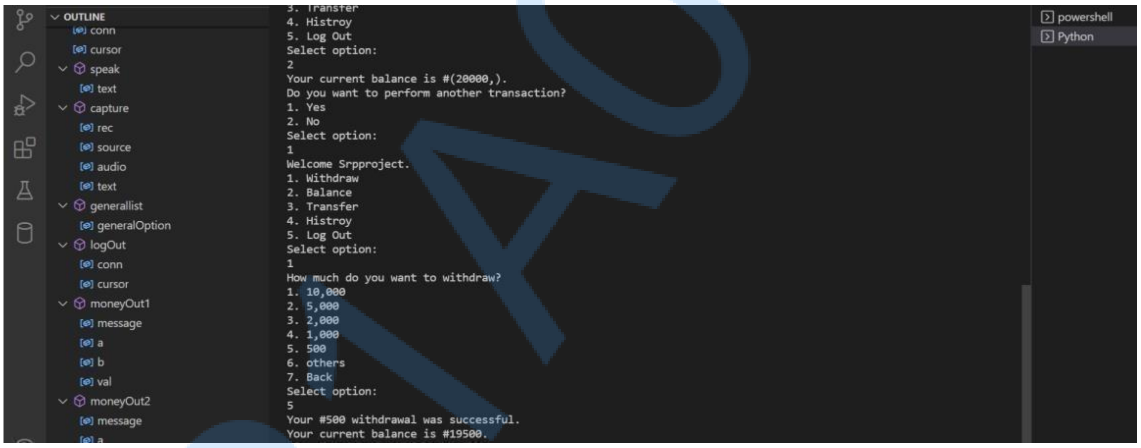
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\eshitha\OneDrive\Documents\pythonprojects> & C:/Users/eshitha/AppData/Local/Programs/Python/Python3
10/python.exe c:/Users/eshitha/OneDrive/Documents/pythonprojects/language.py
Welcome to
1. Login
2. Register
2
2
Enter first name:
Srp
Srp
Enter Last Name:
project
project
Enter your user name:
s_pro
s_pro
Enter your email address:
srp@.com
srp@.com
Enter your pin:
5643
5643
Enter your balance:
20000
20000
('Srp', 'project', 's_pro', 'srp@.com', 5643, 20000)
Welcome Srp project.
Do you wish to transact?
1. Yes
2. No
```

Figure 9: Registration



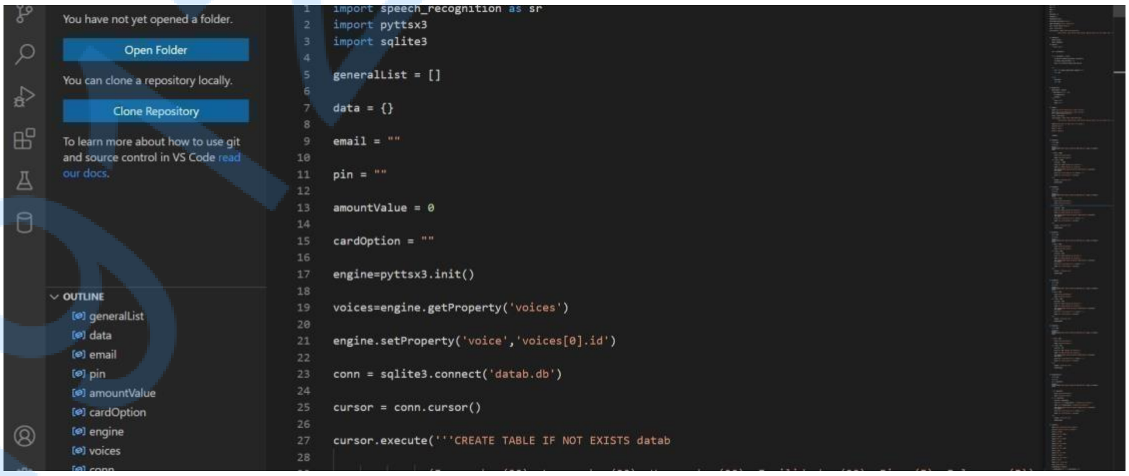
```
20000
20000
('Srp', 'project', 's_pro', 'srp@.com', 5643, 20000)
Welcome Srp project.
Do you wish to transact?
1. Yes
2. No
Select option:
1
Enter your Email:
srp@.com
Enter your Pin:
5643
5643
Welcome Srpproject.
1. Withdraw
2. Balance
3. Transfer
4. Histroy
5. Log Out
Select option:
2
Your current balance is #(20000,).
Do you want to perform another transaction?
1. Yes
2. No
Select option:
1
Welcome Srpproject.
1. Withdraw
2. Balance
3. Transfer
4. Histroy
```

Figure 10: Balance checking



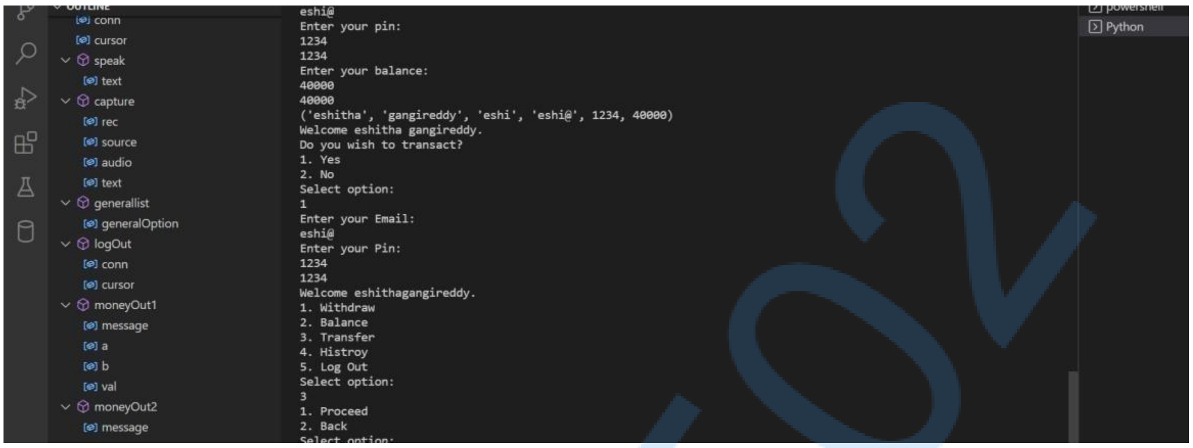
```
3. Transfer
4. Histroy
5. Log Out
Select option:
2
Your current balance is #(20000,).
Do you want to perform another transaction?
1. Yes
2. No
Select option:
1
Welcome Srpproject.
1. Withdraw
2. Balance
3. Transfer
4. Histroy
5. Log Out
Select option:
1
How much do you want to withdraw?
1. 10,000
2. 5,000
3. 2,000
4. 1,000
5. 500
6. others
7. Back
Select option:
5
Your #500 withdrawal was successful.
Your current balance is #19500.
```

Figure 11: Withdrawal


You have not yet opened a folder.
Open Folder
You can clone a repository locally.
Clone Repository
To learn more about how to use git and source control in VS Code read our docs.
OUTLINE
generalList
data
email
pin
amountValue
cardOption
engine
voices
conn

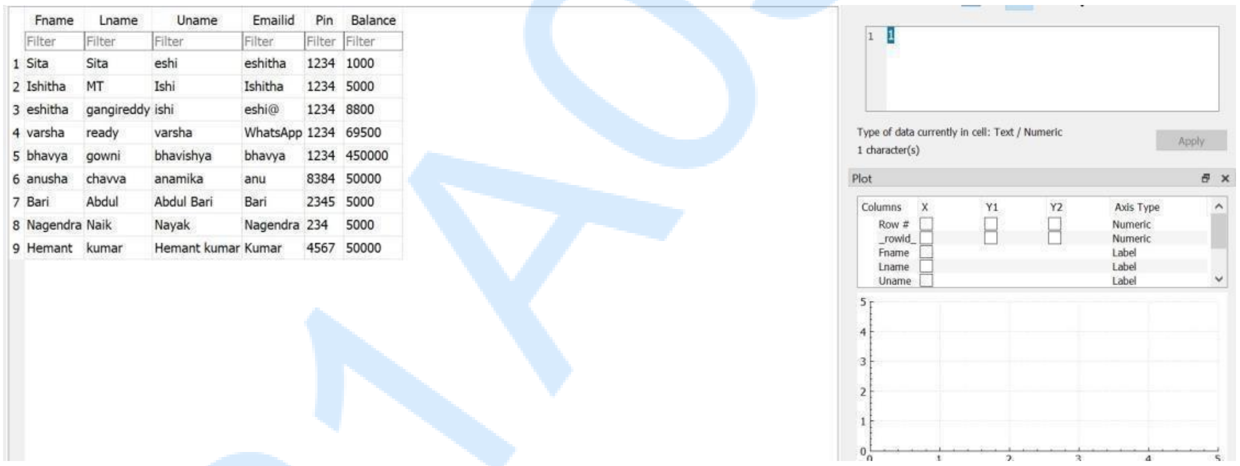
```
import speech_recognition as sr
import pyttsx3
import sqlite3

generalList = []

data = {}

email = ""

pin = ""

amountValue = 0

cardOption = ""

engine=pyttsx3.init()

voices=engine.getProperty('voices')

engine.setProperty('voice','voices[0].id')

conn = sqlite3.connect('datab.db')

cursor = conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS datab
```

Figure 12: Importing modules



```
eshi@
Enter your pin:
1234
1234
Enter your balance:
40000
40000
('eshitha', 'gangireddy', 'eshi', 'eshi@', 1234, 40000)
Welcome eshitha gangireddy.
Do you wish to transact?
1. Yes
2. No
Select option:
1
Enter your Email:
eshi@
Enter your Pin:
1234
1234
Welcome eshithagangireddy.
1. Withdraw
2. Balance
3. Transfer
4. Histroy
5. Log Out
Select option:
3
1. Proceed
2. Back
Select option:
```

Figure 13: Transfer



	Fname	Lname	Uname	Emailid	Pin	Balance
	Filter	Filter	Filter	Filter	Filter	Filter
1	Sita	Sita	eshi	eshitha	1234	1000
2	Ishitha	MT	Ishi	Ishitha	1234	5000
3	eshitha	gangireddy	ishi	eshi@	1234	8800
4	varsha	ready	varsha	WhatsApp	1234	69500
5	bhavya	gowni	bhavishya	bhavya	1234	450000
6	anusha	chavva	anamika	anu	8384	50000
7	Bari	Abdul	Abdul Bari	Bari	2345	5000
8	Nagendra	Naik	Nayak	Nagendra	234	5000
9	Hemant	kumar	Hemant kumar	Kumar	4567	50000

Type of data currently in cell: Text / Numeric
1 character(s)

Figure 14: Updating database

9. References

- *"Union Bank of India unveiled India's first 'Talking ATM'".*
- *"Talking ATMs". www.unionbankofindia.co.in.*
- *Talking ATM Wikipedia.*
- *"State Bank of India launched its first ever 'Talking ATM' in New Delhi".*
- Make Money Talk. RNIB,2011