ATM Simulator (*Using Python and Tkinter GUI*)

A PROJECT REPORT

Submitted by-

Vishal Sharma (25MCI10217)

In partial fulfillment for the award of the degree of

MASTERS OF COMPUTER APPLICATIONS

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Chandigarh University

Mohali, Punjab

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+
Accredited University

BONAFIDE CERTIFICATE

This is to certify that Vishal Sharma of **Chandigarh University** in **University Institute of Computing** have successfully completed the project titled **simulation (or model) of an ATM** in **PYTHON PROGRAMING** as part of his Masters of Computer Application **(MCA-AIML)** in 1st Semester who carried out the project work under my supervision.

External Signature
Signature

(UIC)

Project Supervisor

(UIC)

Index

1. Introduction

The modern world depends heavily on computer-based systems.
Banking is one of the best examples of automation in daily life.
Automatic Teller Machines (ATMs) enable customers to perform simple banking transactions like checking balance, depositing, or withdrawing money, anytime and anywhere.

This project is a **simulation (or model) of an ATM** built using the **Python programming language**.
It demonstrates how user authentication (PIN system), transaction handling, and balance updates can be managed digitally.

Unlike physical ATMs that work on interconnected bank servers, this system runs offline as a teaching and learning tool.
It uses a graphical interface (GUI) so users can click buttons instead of typing commands on the console.

---

2. Objectives

The main objectives of this project are:

1. to design a simple ATM system that allows deposit, withdrawal, and balance inquiries;
2. to implement user authentication using a secure PIN;
3. to use Python's GUI library (**Tkinter**) for user-friendly screens and buttons;
4. to update the customer's balance instantly based on transactions;
5. to demonstrate basic event-driven programming in Python.

This project gives students a real-life idea of how software systems manage financial transactions, securely and efficiently, in a simple coded form.

---

3. Tools and Technologies Used

Component	Description
Programming Language	Python 3.x
GUI Library	Tkinter (standard Python library for graphical user interface)

Component	Description
IDE / Editor	IDLE / VS Code / PyCharm
Operating System	Windows / Linux / macOS
Modules Used	tkinter, messagebox
Concepts Used	Functions, loops, conditional statements, global variables, GUI event handling

---

4. Problem Statement

In a typical bank, customers depend on either cashiers or real ATMs to perform basic operations.
While learning about software systems, students often see textual programs that require typing commands.
However, real-world banking systems use graphical screens and button-based operations.

So the problem was:

How can we simulate the working of a real ATM interface in Python, using GUI elements and secure logic?

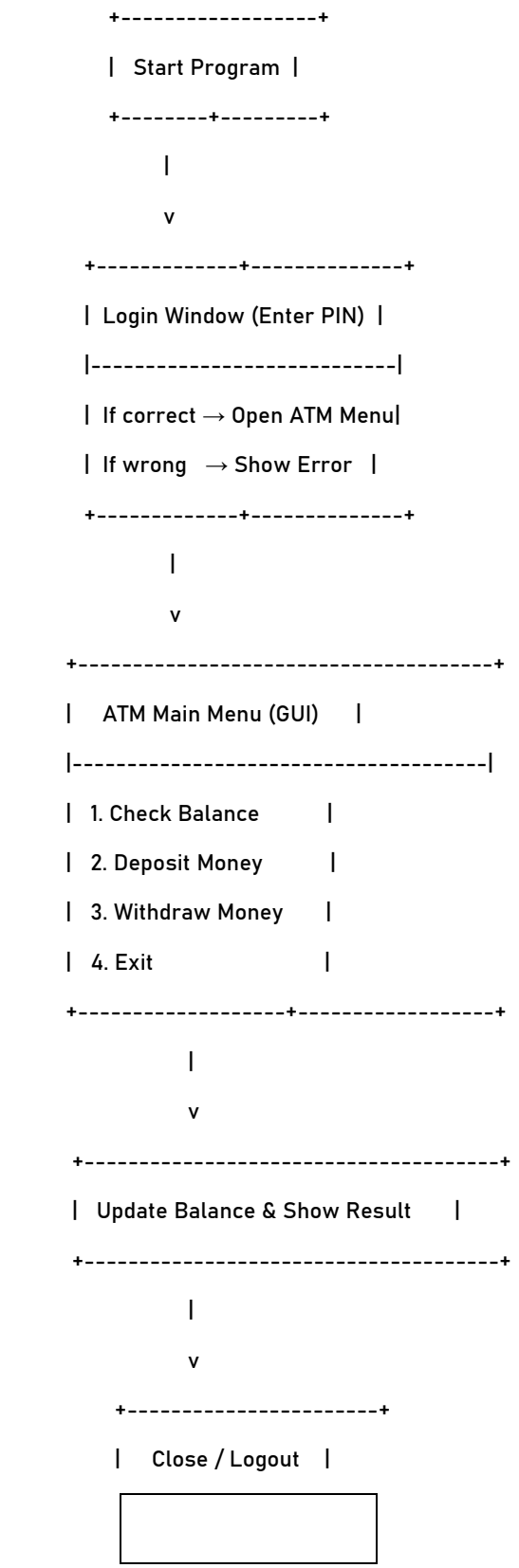
This project solves that by creating a **virtual ATM** on a computer screen that works like a real one, showing current balance and modifying it according to user transactions.

---

5. System Design and Working

5.1 Flow of the System

text



5.2 System Components

1. **Login Screen** – asks user to enter a 4-digit PIN to verify access.
2. **Main ATM Window** – contains:

- Balance display label
- Entry box for amount
- Buttons for *Check Balance*, *Deposit*, *Withdraw*, *Exit*

3. **Functionality** – each button performs a specific function:
    - Balance increases when money is deposited
    - Balance decreases when money is withdrawn
    - Message boxes show transaction success, error, or warnings
4. **Data Handling** – the program stores the user's running balance in a variable and updates it dynamically.

---

6. Implementation Details

The project is developed using **Python Tkinter**, which provides easy tools for creating windows, labels, buttons, and pop-ups.

6.1 Login Window

- User must enter the correct **PIN (1234)** to proceed.
- If entered PIN matches the stored PIN, it shows a success message and opens the main ATM dashboard window.
- Otherwise, it shows an error popup.

6.2 ATM Menu Window

Buttons and functions:

1. **Check Balance:** Displays a popup showing the current balance and updates the label on the screen.
2. **Deposit Money:** Takes the amount typed by the user and adds it to the main balance.
3. **Withdraw Money:** Subtracts the entered amount from the balance (after checking if sufficient funds exist).
4. **Exit:** Closes the ATM window and returns to login.

A **Label** on the top always shows the *live current balance*, which changes immediately after each transaction.

6.3 Main Code Logic (Summary)

Python

```
balance += amt       # increases balance on deposit
balance -= amt       # decreases balance on withdrawal
balance_label.config(text=f"Balance: ₹{balance}")  # refresh display
```

Popup boxes are created using:

Python

```
messagebox.showinfo("Title", "Message")
messagebox.showerror("Error", "Message")
messagebox.showwarning("Warning", "Message")
```

This interactive feedback makes the system easier and more user-friendly.

---

7. Testing and Output

7.1 Testing Cases

Test Case	**Input / Action**	**Expected Output**
Correct PIN	1234	Login successful
Wrong PIN	0000	Error message “Incorrect PIN”
Check Balance	Click Check Balance	“Your current balance is ₹10000”
Deposit Money	Enter 2000	“₹2000 deposited successfully” → balance becomes 12000
Withdraw Money	Enter 1000	“₹1000 withdrawn successfully” → balance becomes 11000
Withdraw More Than Available	Enter 20000	“Insufficient Funds”
Empty amount field	No amount entered	“Please enter an amount!”
Exit	Click Exit	Program closes with a thank-you message

7.2 Sample Screens (Describe / Insert)

1. **Login Window:** “Welcome to Python Bank ATM” – 4-digit PIN input and login button.
2. **Main ATM Menu:** Buttons for Deposit, Withdraw, Check Balance, Exit.

3. **Pop-up Messages:** Confirmation boxes for deposit and withdrawal.
4. **Balance Display:** Label showing " 💰 Current Balance: ₹xxxx.xx", updated live.

---

8. Results and Discussion

After testing, the following results were observed:

- The program successfully validates user login with correct PIN.
- Depositing and withdrawing money updates the on-screen balance instantly.
- Error messages appear correctly for invalid or blank input, ensuring robustness.
- The GUI interface is simple, clear, and easy to use even for a non-technical person.

This shows that Python can easily handle small-scale automation logic using very few lines of code and simple libraries.

---

9. Advantages of the System

1. **User-Friendly:** GUI-based, button-driven design instead of typing commands.
2. **Instant Feedback:** Message boxes confirm every action.
3. **Error Proof:** Handles wrong input gracefully.
4. **Educational Value:** Demonstrates real-world banking simulation in a compact Python program.
5. **Extendable:** Can be improved to store multiple accounts or log transactions using a database.

---

10. Future Scope

This ATM Simulator can be further upgraded to include:

- Multiple user accounts with separate PINs.

- Database connection using SQLite or MySQL for permanent storage.
- Transaction history (mini statement).
- Secure PIN encryption.
- GUI enhancements like account summary pages.
- Receipt or PDF report generation using Python's ReportLab library.

These extensions would make it more similar to a true banking application.

---

11. CODE

```
import tkinter as tk
from tkinter import messagebox


# ----- Account Data -----
stored_pin = "1234"
balance = 10000.0  # Starting balance


# ----- ATM Functions -----
def check_balance():
    """Show the current balance."""
    messagebox.showinfo("Balance", f"Your current balance is ₹{balance:.2f}")
    balance_label.config(text=f"💰 Balance: ₹{balance:.2f}")


def deposit_money():
    """Add money to the current balance."""
    global balance
    amount = amount_entry.get()

    if not amount:
        messagebox.showwarning("Input Error", "Please enter an amount!")
        return
```

```
    try:
        amt = float(amount)
        if amt <= 0:
            messagebox.showwarning("Invalid", "Enter a positive amount!")
        else:
            balance += amt   # <--- Updates the global balance
            messagebox.showinfo("Success",
                                f"₹{amt:.2f} deposited successfully!\nNew Balance: ₹{balance:.2f}")
            balance_label.config(text=f"💰 Balance: ₹{balance:.2f}")  # Update label
            amount_entry.delete(0, tk.END)
    except ValueError:
        messagebox.showerror("Error", "Enter a numeric amount!")


def withdraw_money():
    """Withdraw money from the balance."""
    global balance
    amount = amount_entry.get()

    if not amount:
        messagebox.showwarning("Input Error", "Please enter an amount!")
        return

    try:
        amt = float(amount)
        if amt <= 0:
            messagebox.showwarning("Invalid", "Enter a positive amount!")
        elif amt > balance:
            messagebox.showerror("Insufficient Funds", "You don't have enough balance!")
        else:
            balance -= amt   # <--- Updates the global balance
            messagebox.showinfo("Success",
```

```
                f"₹{amt:.2f} withdrawn successfully!\nRemaining Balance: ₹{balance:.2f}")
            balance_label.config(text=f"💰 Balance: ₹{balance:.2f}")  # Update label
            amount_entry.delete(0, tk.END)
        except ValueError:
            messagebox.showerror("Error", "Enter a numeric amount!")

    def logout():
        messagebox.showinfo("Goodbye", "Thank you for using Python ATM 💳")
        atm_window.destroy()
        login_screen()

    # ----- Login Screen -----
    def login_screen():
        def verify_login():
            pin = pin_entry.get()
            if pin == stored_pin:
                messagebox.showinfo("Welcome", "Login Successful ✅")
                login.destroy()
                main_atm_window()
            else:
                messagebox.showerror("Wrong PIN", "Incorrect PIN. Try again!")

        global login, pin_entry
        login = tk.Tk()
        login.title("Python Bank - ATM Login")
        login.geometry("300x200")
        login.config(bg="#e6f2ff")

        tk.Label(login, text="Welcome to Python Bank ATM 🏦",
                 bg="#e6f2ff", font=("Arial", 12, "bold")).pack(pady=10)
        tk.Label(login, text="Enter your 4-digit PIN:", bg="#e6f2ff").pack()
```

```
    pin_entry = tk.Entry(login, show="*", width=10, font=("Arial", 12))
    pin_entry.pack(pady=5)

    tk.Button(login, text="Login", width=10, bg="#4CAF50", fg="white",
              command=verify_login).pack(pady=10)

    login.mainloop()

# ----- Main ATM Menu Screen -----
def main_atm_window():
    global atm_window, amount_entry, balance_label

    atm_window = tk.Tk()
    atm_window.title("Python Bank ATM")
    atm_window.geometry("400x400")
    atm_window.config(bg="#f2ffe6")

    tk.Label(atm_window, text="ATM Main Menu",
             font=("Arial", 14, "bold"), bg="#f2ffe6").pack(pady=10)

    # Current balance label (updates live)
    balance_label = tk.Label(atm_window,
                             text=f" Balance: ₹{balance:.2f}",
                             font=("Arial", 12, "bold"),
                             bg="#f2ffe6", fg="#333333")
    balance_label.pack(pady=8)

    tk.Label(atm_window, text="Enter amount:", bg="#f2ffe6").pack(pady=5)
    amount_entry = tk.Entry(atm_window, width=15, font=("Arial", 12))
    amount_entry.pack(pady=5)
```

```
    tk.Button(atm_window, text="Check Balance",
              command=check_balance, width=20,
              bg="#2196F3", fg="white").pack(pady=8)
    tk.Button(atm_window, text="Deposit Money",
              command=deposit_money, width=20,
              bg="#4CAF50", fg="white").pack(pady=8)
    tk.Button(atm_window, text="Withdraw Money",
              command=withdraw_money, width=20,
              bg="#FF9800", fg="white").pack(pady=8)

    tk.Button(atm_window, text="Exit", command=logout,
              width=10, bg="#f44336", fg="white").pack(pady=10)

    atm_window.mainloop()

# ----- Run the Program -----
login_screen()
```

---

12. Conclusion

The project *"ATM Simulator Using Python and Tkinter"* successfully demonstrates a simple, interactive, and functional model of a real ATM.
It covers essential programming aspects like authentication, transaction management, and GUI design.
Though small in scale, it represents the concept of **automation, accuracy, and simplicity** in software-based financial systems.

This model can serve as a training or learning tool for students to understand how digital banking applications function internally.
It highlights that with just core Python libraries, one can build practical and user-friendly desktop solutions.

---

13. References

1. Python Official Documentation – https://docs.python.org
2. Tkinter Library Guide – https://tkdocs.com
3. Stack Overflow – Community Discussion Forum
4. Python Crash Course by Eric Matthes (Book)

GITHUB LINK -

https://github.com/vishalsharma15012002/ATM-Stimulator.git