NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY, GREATER NOIDA
(An Autonomous Institute)

BANKING SYSTEM

Python Bank with OTP Verification

A Python Project Submitted

in
Electronics and communication engineering

by

Aakanksha Mishra (Roll no. 2401330310001)
Anshu Sahani (Roll no. 2401330310024)
Goldi Kushwaha (Roll no. 2401330310039)
Ishita Srivastava (Roll no. 2401330310051)
Kaustubh (Roll no. 2401330310055)

Under the Supervision of

Prof. Mr. Yaduvir Singh
(Department of CSE AI)

Prof. Mr. Amit Bindal (Asst. Proff)
(Department of CSE IOT)

INTRODUCTION

Banking System Using Advanced Python

The "Banking System Using Advanced Python" is a simplified version of real-world banking software, developed to perform common banking tasks in a user-friendly way. This project aims to help users manage their accounts easily and securely while giving us hands-on experience in building functional applications using Python.

It includes key banking operations such as:

1. Account Creation
2. Deposit and Withdrawal
3. Fund Transfer Between Accounts
4. Balance Inquiry and Mini Statements

We used advanced Python concepts to make the system more powerful and realistic:

1. Object-Oriented Programming – to organize the code into clean and reusable structures

2. File Handling – to store user data securely

3. Error and Exception Handling – to make the system stable and user-friendly

4. Menu-Driven Interface – for simple and step-by-step navigation

Certificate

This is to certify that
Kaustubh, Aakanksha Mishra, Ishita Srivastava, Goldi Kushwaha, and Anshu Sahani have successfully completed their project titled *"Banking System Using Advanced Python"* under the guidance of Mr. Yaduvir Singh (*Project Guide*) and Mr. Amit Bindal (*Assistant Professor*).

The team demonstrated strong technical skills, teamwork, and creativity in developing a functional and user-friendly banking system.

Date: June 20, 2025

Place: Greater Noida

Signature:

Designation:

Acknowledgement

Guidance & Mentorship

Our deepest appreciation goes to Mr. Yaduvir Singh, *Project Guide*, and Mr. Amit Bindal, *Assistant Professor*. Their clear vision helped us define realistic goals, choose robust design patterns, and adhere to best coding practices.

Institutional Support

We thank the Noida Institute of Engineering and Technology, Greater Noida for:

1. Providing well-equipped labs, high-speed internet, and access to relevant software licenses.

2. Hosting knowledge-sharing events and workshops that sharpened our technical and soft skills alike.

Team Effort

This project stands as proof that together truly is better.

Team Member	Key Contributions
Kaustubh	Core architecture, version-control strategy, API design
Aakanksha Mishra	Requirement analysis, documentation, comprehensive testing
Ishita Srivastava	Database schema design, security implementation
Goldi Kushwaha	User-interface flow, accessibility enhancements
Anshu Sahani	Module integration, performance optimization

Sincerely,

Kaustubh, Aakanksha mishra, Anshu sahani, Ishita srivastava, Goldi kushwaha.

Abstract

Project Title: "Python Bank – A Complete Banking System"

The project "Python Bank" is a complete and secure banking system developed using Advanced Python, designed especially for college-level learning and real-world application. The goal of this project is to simulate a real banking environment where users can perform basic financial operations safely and efficiently through a console-based interface.

This banking system allows users to perform important banking actions such as creating a new account, depositing and withdrawing money, checking their balance, viewing recent transactions, and displaying all account details. The interface is user-friendly, interactive, and backed with strong input validation to ensure that all data entered is correct and secure.

Key Features:

1. Account Creation:
Users can easily create a new account by entering their name and mobile number. This is completely custom-built using Python code without any external framework.

2. Deposit Amount:
Allows users to safely deposit money into their registered accounts after verifying with OTP.

3. Withdraw Amount:
Enables users to withdraw money only after account verification and OTP confirmation to ensure transaction security.

4. Check Balance:
Users can check their current account balance anytime with proper authentication.

5. Display All Accounts:
Admins or users (as per access) can view a list of all created accounts in the system.

6. Input Validation:
Every feature is equipped with input validation. If a user enters invalid data (like wrong account number or mobile), the system displays helpful error messages to guide them.

7. OTP Verification System:
For security, features from point 2 to 6 require the user to input their registered account number and mobile number, followed by an OTP (One-Time Password) sent to the registered mobile. This step adds an extra layer of safety to every transaction.

Technology Used

Python Programming Language:

The entire application is written in Python, utilizing object-oriented programming (OOP) principles. It uses classes like Account, Bank, and BankGUI to manage bank operations, accounts, and user interaction in a modular way.

Tkinter GUI Library:

The graphical user interface is built using Python's built-in Tkinter module. It provides widgets such as Label, Entry, Button, and Toplevel to create forms, buttons, OTP windows, and dialogs. Tkinter’s grid() layout manager is used to structure the UI components cleanly.

OTP Verification with Random Module:

To enhance security, the app uses Python's random module to generate a 6-digit OTP (One-Time Password) before sensitive operations like deposit and withdrawal. The OTP verification ensures that only authorized actions are processed.

Validation & Error Handling:

The code includes built-in input validation for account numbers, names, and numeric fields using string methods and exception handling (try-except). It ensures robust and user-friendly input checking for better reliability.

Message Handling:

The messagebox module from Tkinter is used to provide user feedback through popup dialogs for success, error, and info messages. This makes the interaction more intuitive and helps guide users through the process.

Objective of the Project

The main objective of this project is to design and develop a complete banking system using Python and SQLite that simulates the core operations of a real bank in a console-based environment. The system is meant to provide a clear understanding of how a banking application works—right from account creation to secure transactions—with a focus on security, simplicity, and efficiency.

Key Goals of the Project:

1. To create a menu-driven, interactive banking system where users can perform banking operations with ease.

2. To apply Python programming in a real-life simulation scenario using object-oriented principles, file handling, and input validation.

3. To use SQLite as the backend database for storing account details, transaction history, and user information in a lightweight and efficient manner.

4. To implement a basic security mechanism, including OTP-based verification, to simulate authentication for sensitive operations like deposits, withdrawals, and balance checks.

FUTURE ENHANCEMENTS

The Python Bank System can be further enhanced by adding features like a transaction history log with timestamps, integration of real SMS or email-based OTP for better security, user authentication using login credentials, and a database like SQLite for persistent data storage. Additionally, migrating the GUI to a web-based interface or mobile app and including features like interest calculation or account types (savings, current) can make the system more robust and user-friendly.

Scalable for Future Enhancements:

Though the current version is limited to basic banking features in a console, the system can be expanded in the future with:

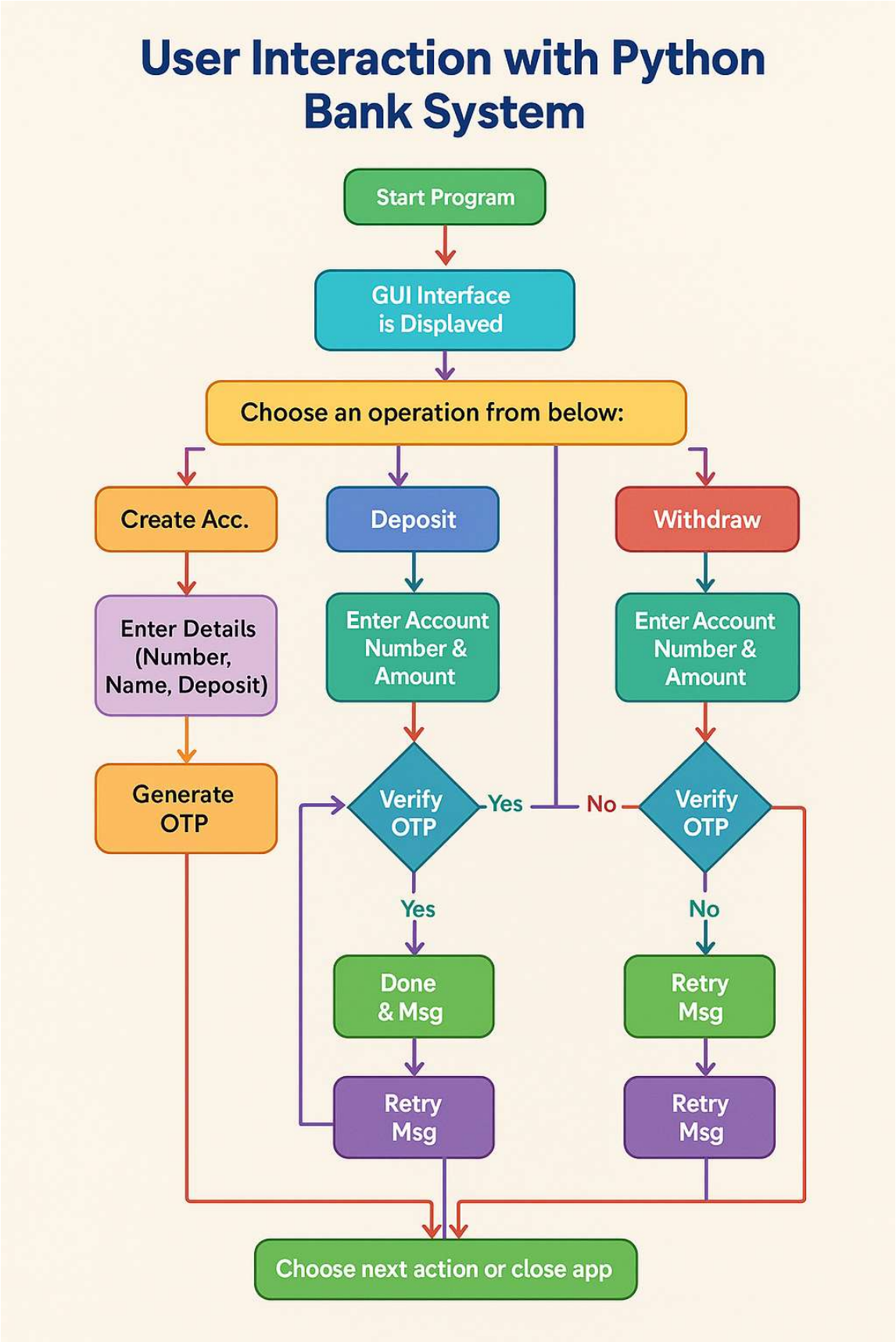
1. GUI using Tkinter or PyQt

2. Web interface using Flask or Django

3. Real OTP verification using SMS APIs

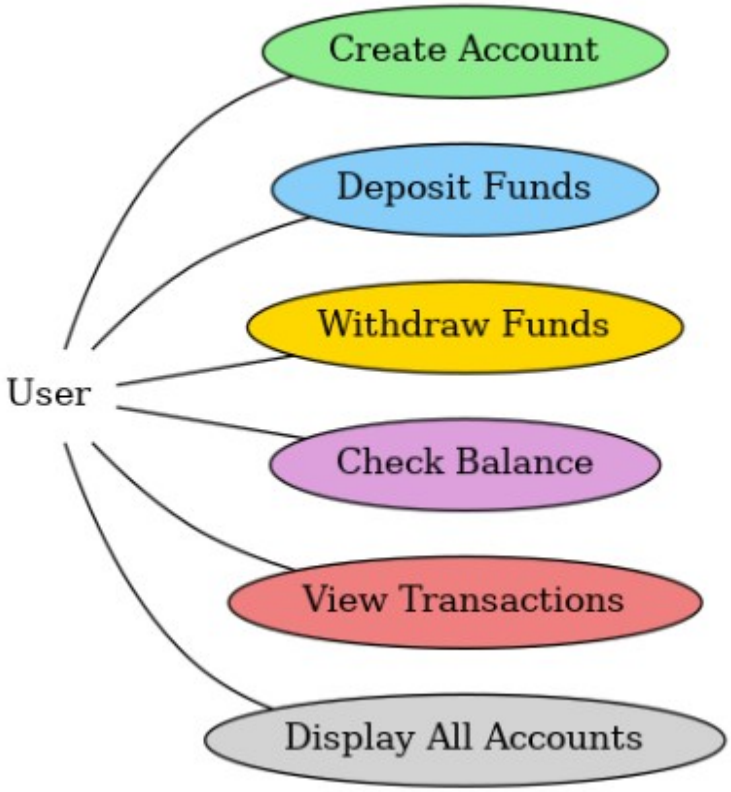
4. Integration with cloud-based databases

1. Flowchart of User Interactions (System Flow)



x

2. Use Case Diagram (Functional Overview)



Green – Creating new accounts

Blue – Depositing funds

Yellow – Withdrawing money

Purple – Checking balance

Red – Viewing recent transactions

Grey – Displaying all accounts

3. Entity-Relationship (ER) Diagram

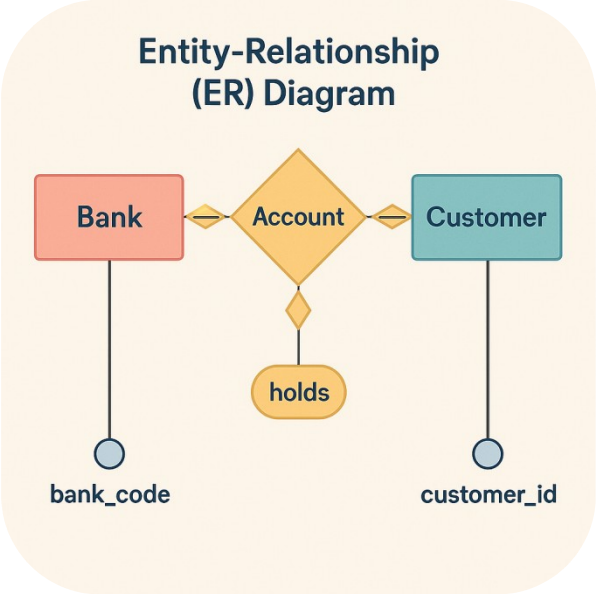


TABLE STRUCTURE:

Table 1: Accounts

Column Name	Data Type	Description	Constraints
acc_no	INTEGER	Unique account number (auto-generated)	Primary Key
name	TEXT	Name of the account holder	Not Null
balance	REAL / FLOAT	Current balance of the account	Default 0.0

Table 2: Transactions

Column Name	Data Type	Description	Constraints
Id	INTEGER	Unique transaction ID (auto-incremented)	Primary Key
Acc_no	INTEGER	Account number related to the transaction	Foreign Key
Type	TEXT	Transaction type: "Deposit" or "Withdraw"	Not Null
Amount	REAL / FLOAT	Amount deposited or withdrawn	Must be ≥ 0
Date	TEXT / DATETIME	Date and time of the transaction	Not Null

MODULE DESCRIPTION:

Module 1: Account Creation

Purpose:
This module allows users to create a brand-new bank account.

Module 2: Deposit and Withdrawal

Purpose:
Handles all money-in and money-out operations of the account.

Deposit Functionality:

1. The user enters their account number and deposit amount.
2. The balance is updated in the accounts table.

3. A new transaction is recorded in the transactions table with type "Deposit".

Withdrawal Functionality:

1. The user enters account number and amount.
2. The system checks if the balance is sufficient.
3. If yes, it deducts the amount and logs the transaction as type "Withdraw".
4. If not, it shows an error message like “Insufficient Balance.”

Module 3: Transaction History Display

Purpose:
Shows the user all past deposit and withdrawal records.

Module 4: Balance Check

Purpose:
Lets users check their current account balance instantly.

Conclusion

This project, Python Bank System with OTP Verification, has been a comprehensive implementation of core banking functionalities using advanced Python concepts and GUI development with Tkinter. Through this project, we successfully built a system that allows users to create accounts, deposit and withdraw funds with OTP verification, check balances, and view all existing accounts. We incorporated real-time input validation and simulated OTP-based authentication to make the system more secure and interactive.

1. **What we achieved:**
Key features like account creation, deposit, withdrawal, balance checking, and secure OTP verification were implemented with appropriate validations and logical structure. The Bank and Account class structures helped encapsulate the functionalities in a clean, object-oriented manner. The GUI provided a practical front-end experience while simulating banking operations with a strong backend logic.

2. **What we learned:**
This project deepened our understanding of object-oriented programming (OOP) in Python and reinforced GUI design principles using the Tkinter library. We learned how to simulate security measures such as OTP verification, which can be extended to real-world systems. Through rigorous validation handling, we became proficient at managing user input errors and logical exceptions.

References

1. Python Documentation: https://docs.python.org/3/

2. Tkinter GUI Programming:
https://docs.python.org/3/library/tkinter.html

3. GeeksforGeeks: https://www.geeksforgeeks.org/python-gui-tkinter/

4. W3Schools Python Tutorials: https://www.w3schools.com/python/

5. TutorialsPoint Python OOP Concepts:
https://www.tutorialspoint.com/python/python_classes_objects.htm

https://youtu.be/mlIUKyZIUUU?si=5rAQGTpMOg36j_GV

5. Real Python: https://realpython.com/

6. Stack Overflow: https://stackoverflow.com/

7. Python OTP Implementation Concepts: https://pypi.org/project/pyotp/

8. ChatGPT (for ER Diagrams and Use Case Diagrams):
https://chatgpt.com/s/m_6853dd56ddd48191821627b7efc4c5ad

9. Youtube: https://youtu.be/MGhgq-hVlAU?si=BrxQi4jttYUYPHdi

10. Black Box(Fixing Errors): https://www.blackbox.ai/

Appendices

Appendix A – Source Code Overview

- File Name: Banking_System.py(Link)

- Technologies Used: Python 3, Tkinter (GUI), Random module

- Main Components:

  - Account Class: Handles account data and basic operations like deposit, withdraw, and balance check.
  - Bank Class: Manages multiple accounts.
  - BankGUI Class: Provides a graphical user interface for user interactions with features like:
    - Account Creation
    - Deposit (with OTP Verification)
    - Withdrawal (with OTP Verification)
    - Balance Inquiry
    - Display All Accounts

- OTP verification is simulated within the application for educational/demo purposes.

Appendix B – OTP Generation Function

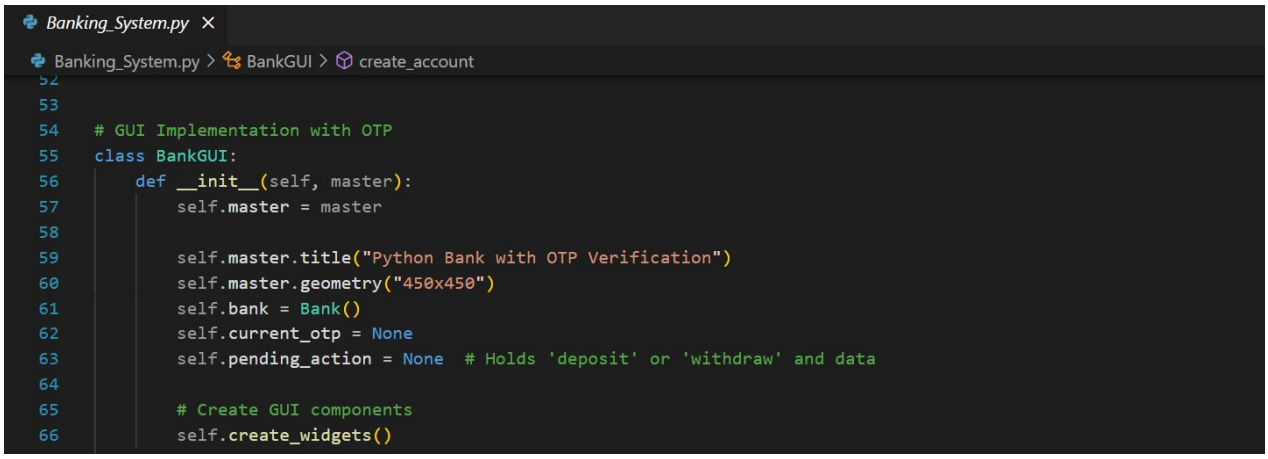
Banking_System.py

Banking_System.py > BankGUI > create_account

```
 55  s BankGUI:
196  def initiate_withdraw(self):
223
224      # Generate OTP
225      self.current_otp = f"{random.randint(100000, 999999)}"
226      self.pending_action = ("withdraw", account, amount)
227      messagebox.showinfo("OTP Sent", f"Your OTP is: {self.current_otp}\n(For demo purposes, OTP is shown here.)")
228      self.prompt_otp()
229
230  def prompt_otp(self):
231      self.otp_win = tk.Toplevel(self.master)
232      self.otp_win.title("OTP Verification")
233      self.otp_win.configure(bg="#e8f0fe")
234      self.otp_win.grab_set() # Make the OTP window modal
235
236      tk.Label(self.otp_win, text="Enter OTP:", font=("Segoe UI", 10, "bold"), bg="#e8f0fe").pack(padx=10, pady=5)
237      self.otp_entry = tk.Entry(self.otp_win, font=("Segoe UI", 10))
238      self.otp_entry.pack(padx=10, pady=5)
239      tk.Button(self.otp_win, text="Verify", command=self.verify_otp, font=("Segoe UI", 10, "bold"),
240                bg="#4CAF50", fg="white").pack(pady=10)
241      self.otp_win.protocol("WM_DELETE_WINDOW", self.on_otp_window_close) # Handle closing the OTP window
242
243  def on_otp_window_close(self):
244      # Reset pending action if OTP window is closed without verification
245      self.current_otp = None
246      self.pending_action = None
247      self.otp_win.destroy()
248
```

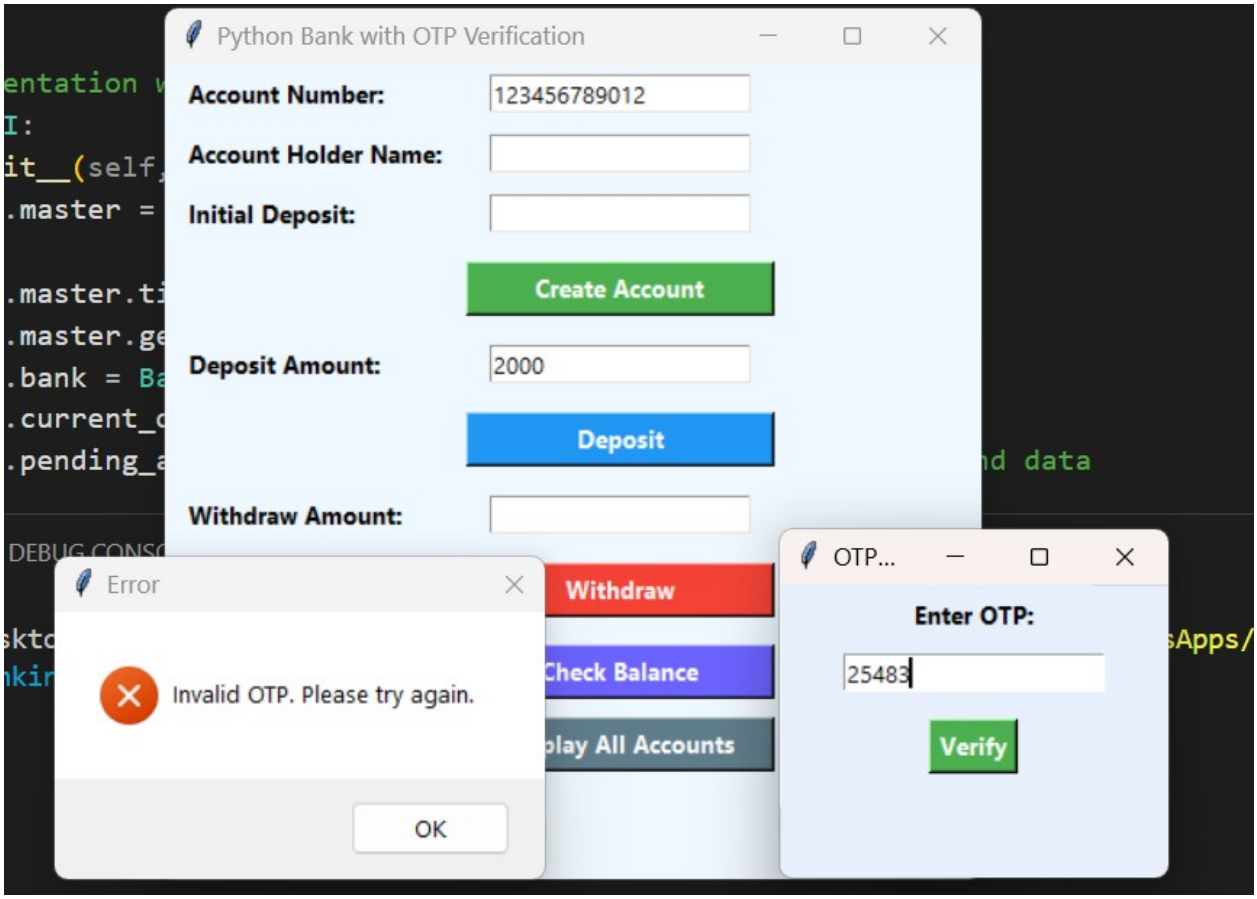
Appendix C – GUI Widgets Layout

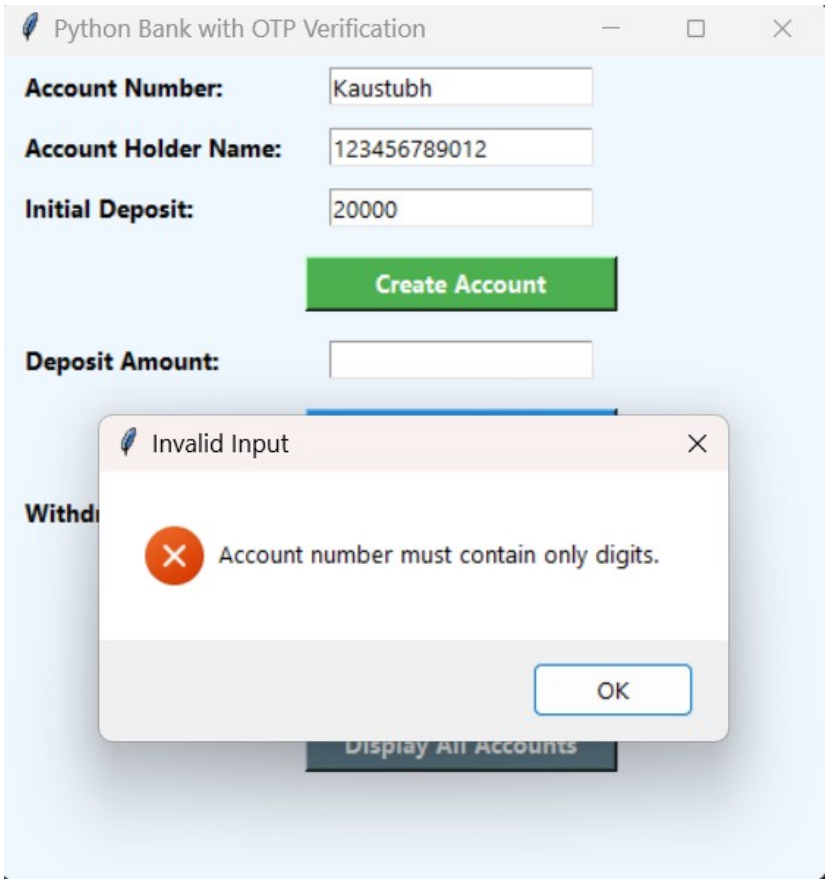
Feature	Widgets Used
Account Creation	Label, Entry, Button
Deposit/Withdraw	Entry, Button, OTP Popup
Check Balance	Button, Messagebox
Display All Accounts	Button, Messagebox

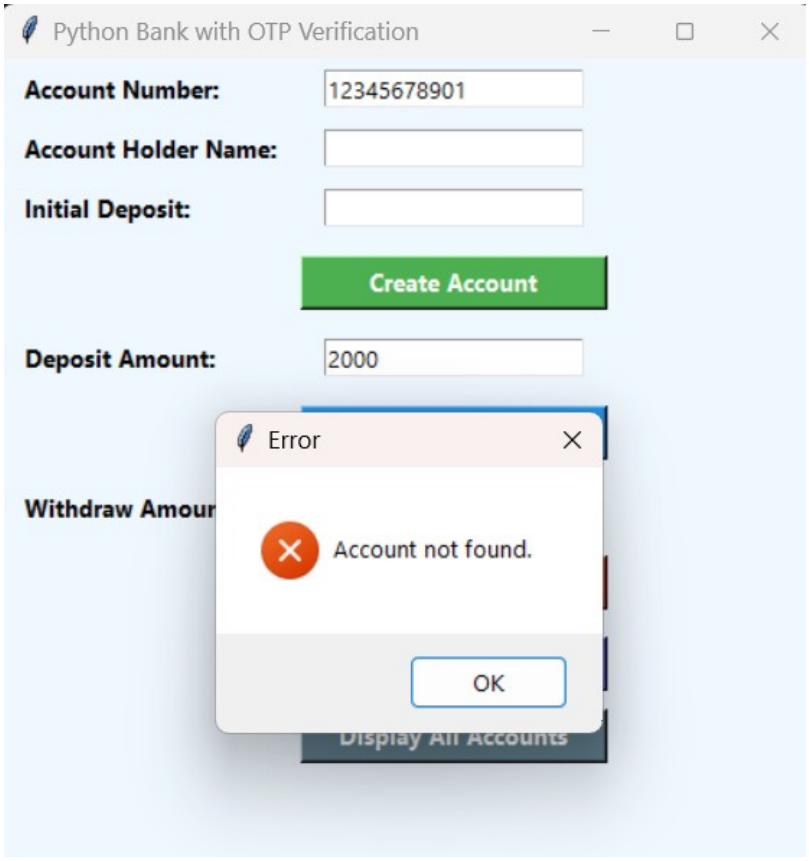


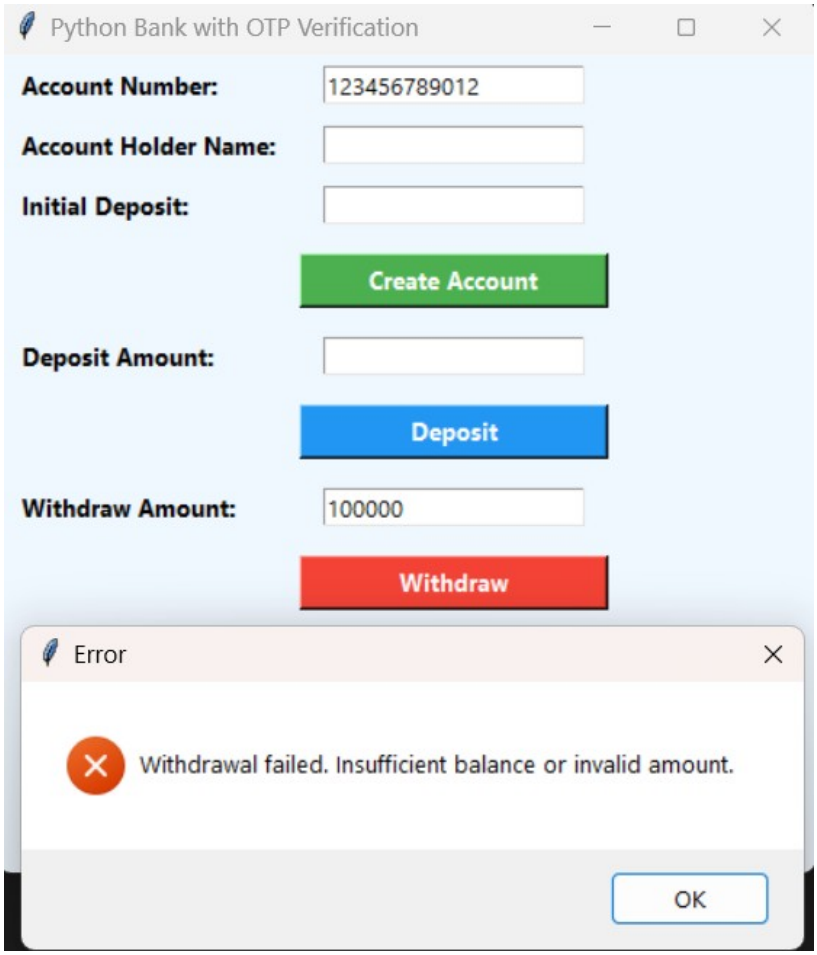
Appendix D – Error Handling Techniques

- Error handling has been incorporated to improve user experience:
- Validation for empty fields, incorrect formats, and invalid numbers.
- Exceptions like ValueError and TclError are handled when parsing input.
- OTP failure is safely managed with message alerts.


Python Bank with OTP Verification
Account Number:
123456789012
Account Holder Name:
Initial Deposit:
Create Account
Deposit Amount:
2000
Deposit
Withdraw Amount:
Withdraw
Check Balance
Display All Accounts
Error
Invalid OTP. Please try again.
OK
OTP...
Enter OTP:
25483
Verify

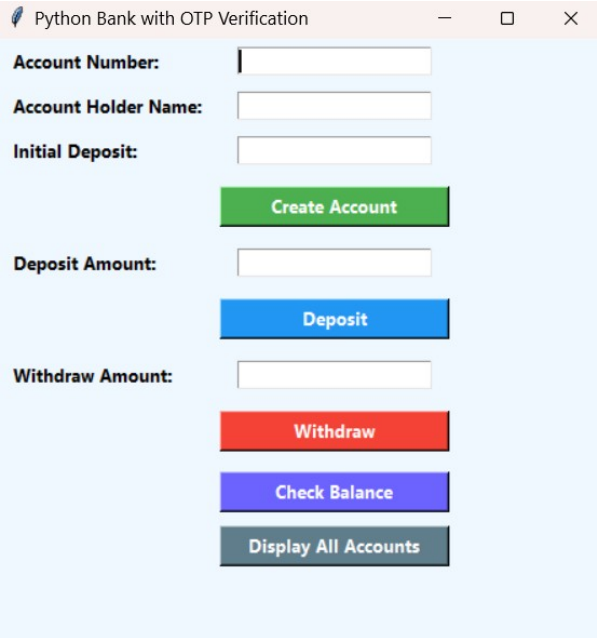

Python Bank with OTP Verification
Account Number:
Kaustubh
Account Holder Name:
123456789012
Initial Deposit:
20000
Create Account
Deposit Amount:
Invalid Input
Account number must contain only digits.
OK


Python Bank with OTP Verification
Account Number:
12345678901
Account Holder Name:
Initial Deposit:
Create Account
Deposit Amount:
2000
Withdraw Amour
Error
Account not found.
OK
Display All Accounts

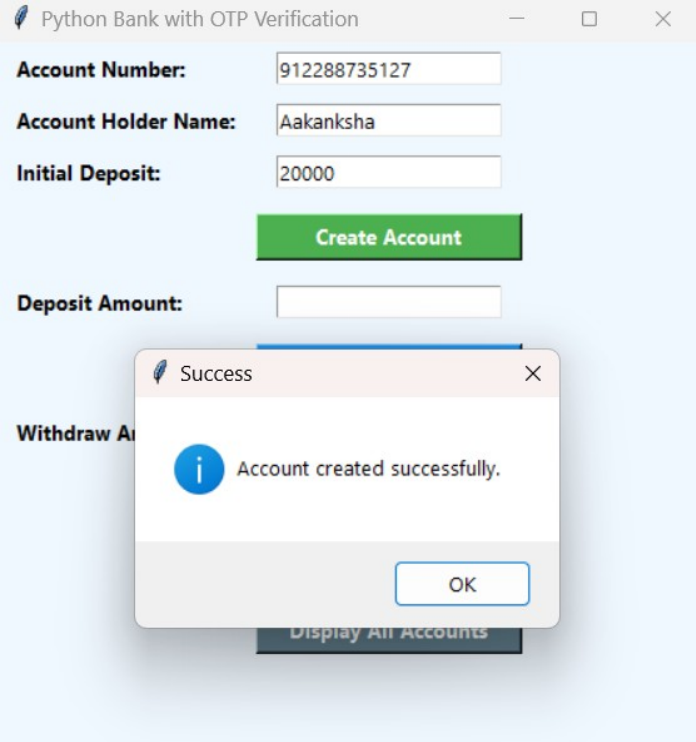

Python Bank with OTP Verification
Account Number:
123456789012
Account Holder Name:
Initial Deposit:
Create Account
Deposit Amount:
Deposit
Withdraw Amount:
100000
Withdraw
Error
Withdrawal failed. Insufficient balance or invalid amount.
OK

Appendix E – Screenshots of Application

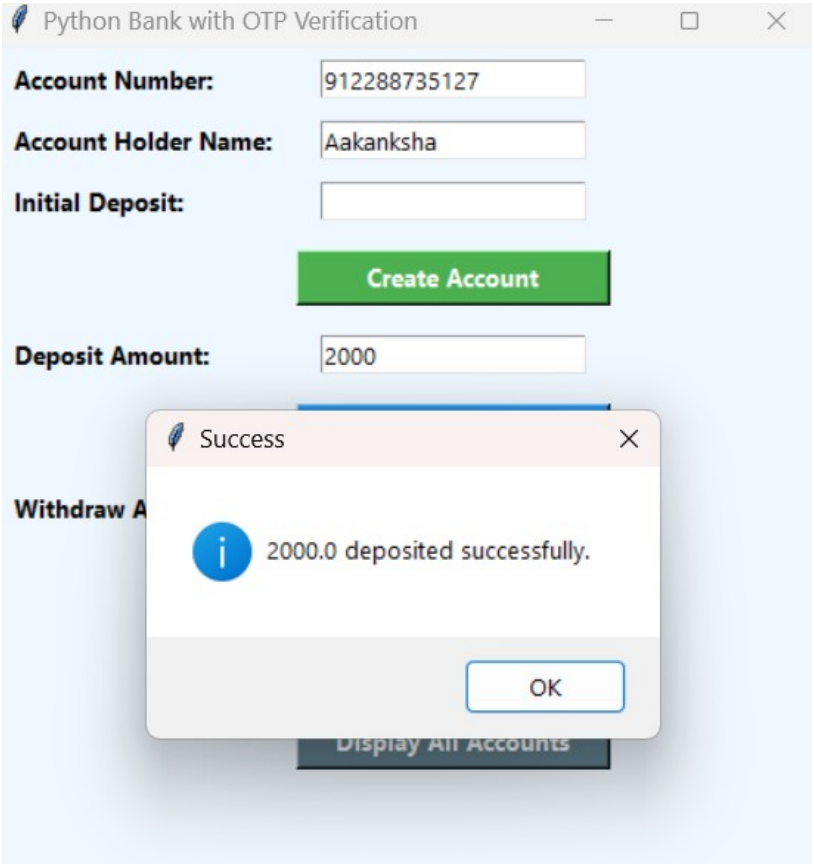
1. Home window (GUI)

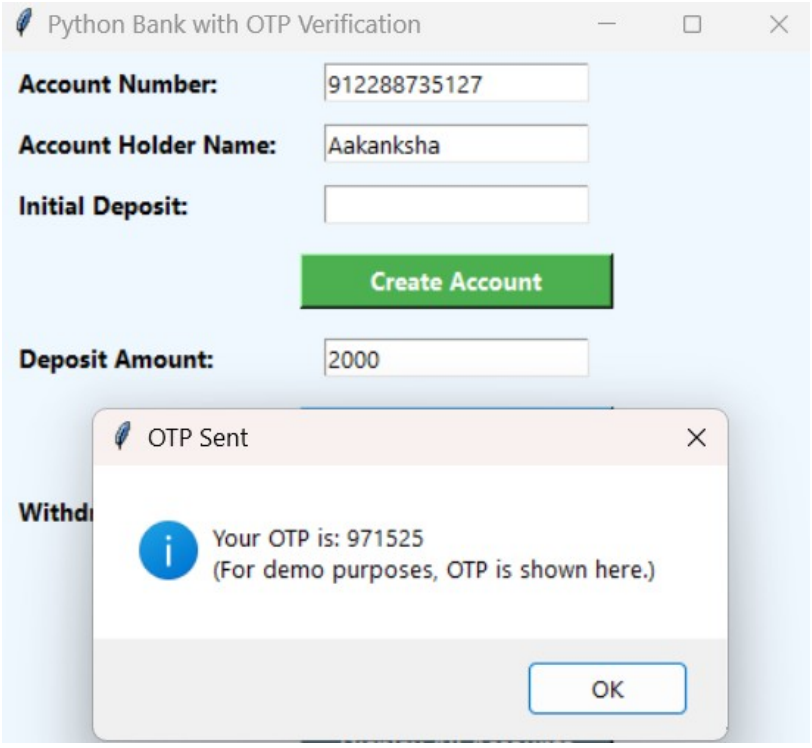


2. Account creation screen

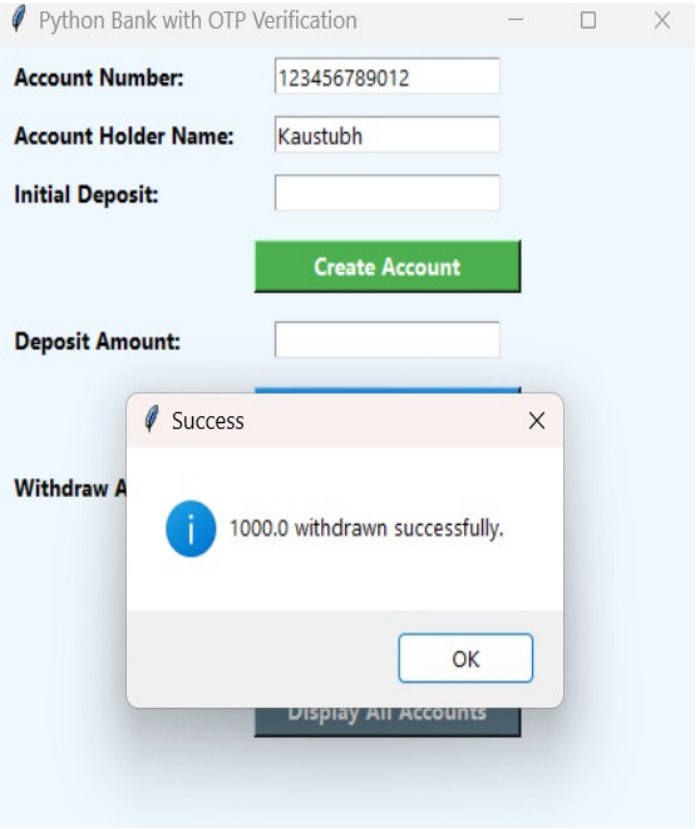


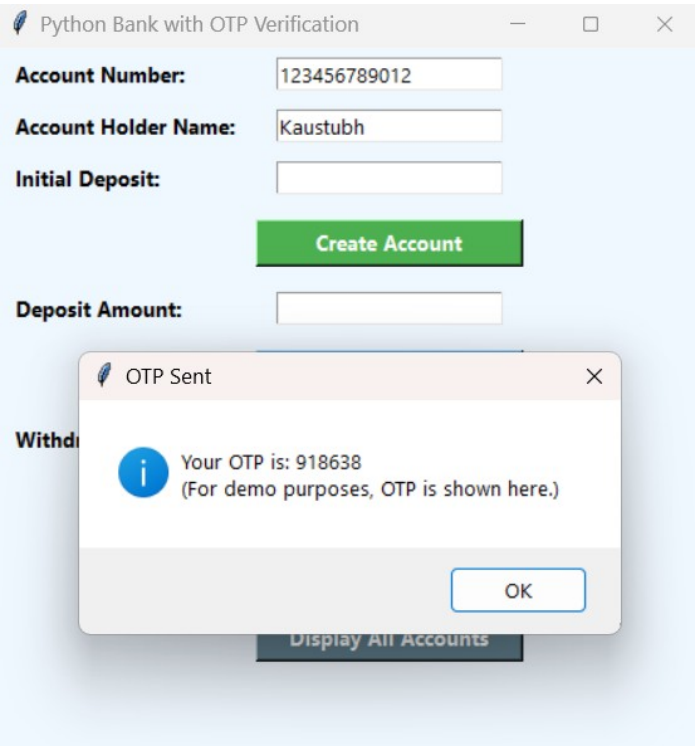
3. Deposit with OTP prompt



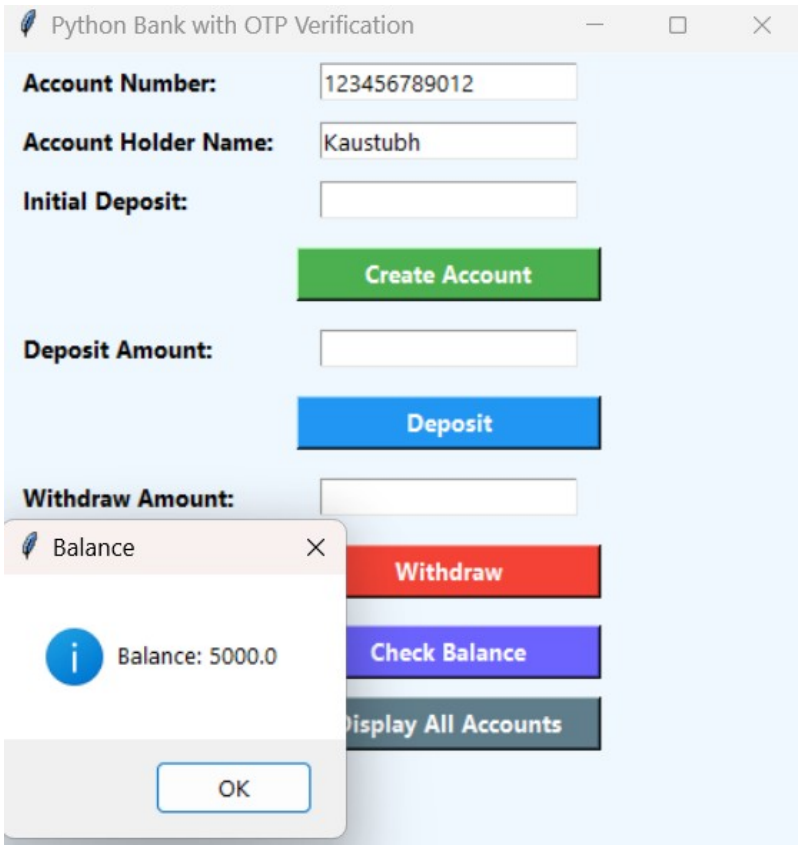


4. Withdraw with OTP prompt





5. Successful balance check



6. Display of all accounts