Simple ATM System - Project Documentation

1. Cover Page

Project Title: Simple ATM System
Project Type: Command-Line Application (Educational Project)
Version: 1.0
Date: November 23, 2025
Prepared By: Development Team

Objective: To create a secure, interactive ATM system simulation that enables users to perform basic banking transactions (withdraw, deposit, check balance) with PIN-based authentication.

---

2. Introduction

The Simple ATM System is a Python-based command-line application that simulates a real-world ATM interface. It provides users with a secure way to perform essential banking operations including cash withdrawals, deposits, and balance inquiries. The system prioritizes security through PIN verification before any sensitive transaction and demonstrates core programming concepts such as conditionals, loops, input validation, and state management.

Target Users: - Banking and finance students - Beginners learning Python programming - Educational institutions teaching software engineering - Anyone interested in ATM system simulation

Project Scope: Educational demonstration of a basic ATM system with in-memory data storage and PIN-based access control.

---

3. Problem Statement

Current Challenge: Users need a reliable, secure method to access banking services without visiting physical bank branches. Manual transaction processing is error-prone and time-consuming. There is a need for an automated system that allows users to: - Withdraw cash securely - Deposit funds safely - Check account balances instantly - Prevent unauthorized access through authentication

Solution Proposed: An interactive command-line ATM application that provides essential banking functions with PIN verification, input validation, and real-time balance updates.

---

4. Functional Requirements

Req ID	Feature	Description
FR-1	PIN Verification	System verifies user PIN before any transaction
FR-2	Withdraw Cash	Users can withdraw specified amounts if sufficient balance exists
FR-3	Deposit Funds	Users can deposit money to increase account balance
FR-4	Check Balance	Users can view current account balance
FR-5	Menu Navigation	System provides menu-driven interface for ease of use
FR-6	Error Handling	System handles invalid inputs and insufficient funds
FR-7	Exit Functionality	Users can safely exit the application

5. Non-Functional Requirements

Req ID	Requirement	Description
NFR-1	Security	PIN protection for all sensitive operations
NFR-2	Usability	Simple, intuitive menu requiring minimal training
NFR-3	Performance	Instant response for all transactions
NFR-4	Reliability	Consistent behavior across all operations
NFR-5	Data Validation	Input validation to prevent invalid transactions
NFR-6	User Experience	Clear messages and confirmations for all actions

6. System Architecture

6.1 Layered Architecture

```
┌──────────────────────────────────┐
│   User Interface Layer           │
│   (CLI Menu & Prompts)           │
└────────────────┬─────────────────┘
                 │
┌────────────────▼─────────────────┐
│   Input Processing Layer         │
│   (User Choice & PIN Input)      │
└────────────────┬─────────────────┘
                 │
┌────────────────▼─────────────────┐
│   Business Logic Layer           │
│   ├─ verify_pin()                │
│   ├─ withdraw_money()            │
│   ├─ deposit_money()             │
│   └─ check_balance()             │
└────────────────┬─────────────────┘
                 │
┌────────────────▼─────────────────┐
│   Data Management Layer          │
│   ├─ balance                     │
│   └─ PIN storage                 │
└────────────────┬─────────────────┘
                 │
┌────────────────▼─────────────────┐
│   Output Display Layer           │
│   (Confirmation & Error Msgs)    │
└──────────────────────────────────┘
```

6.2 Architecture Components

Layer 1: User Interface - Menu display with numbered options - Text-based prompts for user interaction - Emoji indicators for visual feedback

Layer 2: Input Processing - Captures user menu choice - Routes to appropriate function - Handles PIN input

Layer 3: Business Logic - PIN verification function - Transaction processing functions - Balance update logic - Input validation

Layer 4: Data Management - Initial balance storage - PIN storage - Transaction state management

Layer 5: Output Display - Transaction confirmations - Error messages - Balance display

---

7. Design Diagrams

7.1 Use Case Diagram

```
            ┌─── Withdraw Cash
            │
    ┌───────┴─────────┐
    │  ATM System     │
    └───────┬─────────┘
            │
            ├─── Deposit Funds
            │
            ├─── Check Balance
            │
            └─── Exit System

Authentication:
All operations require PIN verification
```

7.2 Workflow Diagram

```
START
  ↓
Display ATM Menu
  ↓
User Selects Option (1-4)
  │
  ├─→ Option 1: Withdraw
  │    ├─ Verify PIN
  │    ├─ Input Amount
  │    ├─ Check Balance
  │    ├─ Deduct Amount
  │    └─ Display Confirmation
  │
  ├─→ Option 2: Deposit
  │    ├─ Verify PIN
  │    ├─ Input Amount
  │    ├─ Add to Balance
  │    └─ Display Confirmation
  │
  ├─→ Option 3: Check Balance
  │    ├─ Verify PIN
  │    └─ Display Balance
  │
  ├─→ Option 4: Exit
  │    └─ Display Goodbye Message → END
  │
  └─→ Invalid Choice
       └─ Display Error → Loop Back
  ↓
Continue or Exit
```

7.3 Sequence Diagram

```
User       Menu       PIN Check      Transaction      Balance
 │          │            │              │                │
 ├─ Select ─→│            │              │                │
 │ Option 1 │            │              │                │
 │          ├─ Verify ──→│              │                │
 │          │ PIN        │              │                │
 │          │←─ Valid ───│              │                │
 │          ├─ Get Amount ───→             │              │
 │          │←─ Amount   ───│              │              │
 │          │             │─ Check ───────→│              │
 │          │             │←─ Balance ────│              │
 │          │             ├─ Deduct ───────→             │
 │          │←─ Confirm ──│              │               │
 │←─ Display ─│             │              │              │
 │ Result   │             │              │               │
```

7.4 Class/Component Diagram

```
┌─────────────────────────────┐
│    ATM System               │
├─────────────────────────────┤
│ Global Variables:           │
│  - balance: float           │
│  - PIN: string              │
├─────────────────────────────┤
│ Functions:                  │
│  - verify_pin()             │
│  - withdraw()               │
│  - deposit()                │
│  - check_balance()          │
│  - main() [menu loop]       │
└─────────────────────────────┘
         │
         └─→ Transaction
              ├─ Amount: float
              ├─ Type: string
              └─ Status: bool
```

8. Design Decisions & Rationale

8.1 PIN Verification Function

Decision: Create separate verify_pin() function called before transactions.

Rationale: - Separates security logic from business logic - Ensures consistency across all operations - Easy to modify PIN verification rules - Clear authorization checkpoint

8.2 Global Variables for State

Decision: Use global variables for balance and PIN.

Rationale: - Simplicity for educational demonstration - Easy to understand program flow - Maintains state across loop iterations - Direct access without complex structures

8.3 Command-Line Interface

Decision: Text-based menu instead of GUI.

Rationale: - Cross-platform compatibility - No additional dependencies - Easier to understand for beginners - Focuses on core logic rather than UI

8.4 Float Data Type for Currency

Decision: Use Python float for monetary amounts.

Rationale: - Supports decimal values (cents) - Natural representation of money - Easy mathematical operations - Note: In production, Decimal type would be preferred

8.5 Menu-Driven Loop

Decision: Use infinite while loop with break condition.

Rationale: - Standard ATM interaction pattern - Allows multiple transactions in one session - User-controlled exit - Simple to implement and understand

---

9. Implementation Details

9.1 Initial Variables

```
balance = 10000      # Initial account balance
PIN = "1234"         # Default PIN for testing
```

9.2 Core Functions

verify_pin(): - Prompts user for PIN input - Compares with stored PIN - Returns True/False - Displays error message if incorrect

Main Loop: - Displays menu options - Gets user choice - Routes to appropriate function - Continues until user exits

Withdraw Function: - Verifies PIN - Gets withdrawal amount - Checks sufficient balance - Updates balance - Displays confirmation

Deposit Function: - Verifies PIN - Gets deposit amount - Adds to balance - Displays confirmation

Check Balance Function: - Verifies PIN - Displays current balance

9.3 Control Flow

```
Main Loop
├─ Display Menu
├─ Get Choice
├─ If "1" → Verify PIN → Withdraw
├─ If "2" → Verify PIN → Deposit
├─ If "3" → Verify PIN → Check Balance
├─ If "4" → Exit
└─ Else → Error Message
```

---

10. Screenshots / Results

10.1 Initial Menu Display

```
===== ATM MENU =====
1. Withdraw
2. Deposit
3. Check Balance
4. Exit

Enter your choice (1-4):
```

10.2 Successful Withdrawal

```
Enter your PIN: 1234
Enter amount to withdraw: 2000

□ Withdrawal Successful! Amount withdrawn: 2000
□ Available Balance: 8000
```

10.3 Insufficient Funds Error

```
Enter your PIN: 1234
Enter amount to withdraw: 15000

□ Insufficient Balance!
□ Available Balance: 10000
```

10.4 Successful Deposit

```
Enter your PIN: 1234
Enter amount to deposit: 5000

□ Deposit Successful! Amount deposited: 5000
□ Available Balance: 15000
```

10.5 Check Balance

```
Enter your PIN: 1234

□ Your Current Balance: 10000
```

10.6 Incorrect PIN

```
Enter your PIN: 0000

□ Incorrect PIN!
```

---

11. Testing Approach

11.1 Unit Testing

Test Case	Scenario	Expected Result	Status
TC-1	Correct PIN entry	PIN accepted, proceed	Pass
TC-2	Incorrect PIN entry	Error shown, retry	Pass
TC-3	Withdraw valid amount	Balance updated, confirmation	Pass
TC-4	Withdraw more than balance	Error shown, balance unchanged	Pass
TC-5	Deposit valid amount	Balance updated, confirmation	Pass
TC-6	Check balance with correct PIN	Balance displayed	Pass
TC-7	Check balance with wrong PIN	Error shown	Pass
TC-8	Invalid menu choice	Error message, menu repeats	Pass
TC-9	Exit option	Program terminates	Pass
TC-10	Negative withdrawal amount	Handled gracefully	Pass

11.2 Integration Testing

1. **Complete Transaction Cycle:**
   - Enter menu → Verify PIN → Withdraw → Check balance → Exit
2. **Multiple Transactions:**
   - Deposit → Withdraw → Check Balance → Deposit again → Exit
3. **Error Recovery:**
   - Wrong PIN → Retry with correct PIN → Proceed
   - Insufficient funds → Try smaller amount → Success

---

12. Challenges Faced

12.1 Challenge: Input Validation

Issue: Non-numeric input for amount causes program crash.

Solution: Implement try-except blocks for float conversion:

```python
try:
    amount = float(input("Enter amount: "))
except ValueError:
    print("Invalid input. Please enter a number.")
```

12.2 Challenge: Preventing Negative Withdrawals

Issue: User could enter negative amount to increase balance fraudulently.

Solution: Add validation:

```python
if amount <= 0:
    print("Amount must be positive!")
    return
```

12.3 Challenge: Global Variable Management

Issue: Global variables can make code harder to maintain.

Solution: Use global keyword explicitly and document all global variables.

12.4 Challenge: User Experience with PIN

Issue: PIN appears on screen, reducing security feel.

Solution: Could use getpass module in production version.

12.5 Challenge: Float Precision with Currency

Issue: Floating-point arithmetic can cause precision errors.

Solution: In production, use Decimal type for currency values.

---

13. Learnings & Key Takeaways

13.1 Programming Concepts Demonstrated

- **Control Flow:** While loops, if-elif-else statements
- **Functions:** Modular code with reusable components
- **Input/Output:** User interaction and display formatting
- **Data Types:** Working with strings and floats
- **Conditionals:** Complex decision logic

13.2 Software Engineering Principles

- **Security:** PIN verification for access control
- **Validation:** Input validation before processing
- **User Experience:** Clear messages and confirmations
- **Error Handling:** Graceful handling of edge cases
- **Code Organization:** Separation of concerns

13.3 Best Practices Applied

- Meaningful variable names
- Code comments for clarity
- Consistent formatting and structure
- Input validation before processing
- User-friendly error messages
- Proper exit mechanisms

13.4 Real-World ATM Concepts

- Authentication (PIN)
- Transaction processing
- Balance management
- Error scenarios
- User interface design

---

14. Future Enhancements

14.1 Phase 2 Features

- **Account Management:** Multiple user accounts with separate PINs
- **Transaction History:** Record and display transaction logs

- **PIN Change:** Allow users to change their PIN
- **Mini Statement:** Show last 5 transactions
- **Account Overdraft:** Support negative balance with interest

14.2 Phase 3 Features

- **Database Integration:** Persistent storage of accounts
- **Network Banking:** Connect to actual banking system
- **Mobile Interface:** Responsive web interface
- **Security Features:** Encryption, session timeout, account lockout
- **Admin Panel:** Manage accounts and generate reports

14.3 Technical Improvements

- Object-Oriented Design with Account and ATM classes
- Unit testing with pytest framework
- SQLite database for account storage
- Environment variables for secure PIN storage
- Logging system for transaction audit trails
- API endpoints for remote access

14.4 User Experience Enhancements

- GUI with tkinter or PyQt
- Receipt printing simulation
- Multi-language support
- Accessible interface for disabled users
- Transaction confirmation screens
- Card validation simulation

---

15. References

15.1 Python Documentation

- Python Official Docs: https://docs.python.org/3/
- Built-in Functions: https://docs.python.org/3/library/functions.html
- Input/Output: https://docs.python.org/3/tutorial/inputoutput.html

15.2 Banking Concepts

- ATM System Design Patterns
- Security Best Practices in Banking
- Transaction Processing Fundamentals

15.3 Learning Resources

- Real Python Tutorials: https://realpython.com/
- W3Schools Python: https://www.w3schools.com/python/
- GeeksforGeeks: https://www.geeksforgeeks.org/python-tutorial/

15.4 Related Technologies

- SQLite Database: https://www.sqlite.org/
- Flask Web Framework: https://flask.palletsprojects.com/
- Python unittest Framework: https://docs.python.org/3/library/unittest.html

---

Appendix

A. Complete Code Summary

The program consists of: - Initial setup with global variables - 1 PIN verification function - 1 main menu loop - 4 transaction handling branches

B. Installation & Usage

1. Save code as `atm_system.py`
2. Run: `python atm_system.py`
3. Enter PIN: 1234
4. Select options 1-4
5. Exit when done

C. Glossary

Term	Definition
PIN	Personal Identification Number for authentication
Balance	Current account funds available
Withdraw	Remove funds from account
Deposit	Add funds to account
Transaction	Banking operation (withdraw/deposit)
Authentication	Process of verifying user identity
CLI	Command-Line Interface

Document Version: 1.0
Created: November 23, 2025
Status: Complete
File Type: PDF Documentation

End of Document