Report – Python ATM Interface

1. Introduction

The ATM Interface project is a command-line application developed in Python that simulates the core functionalities of a real-world Automated Teller Machine (ATM). The project is designed for educational purposes, providing hands-on experience with user authentication, file-based data persistence, input validation, and modular programming in Python. By building this project, users gain practical insight into how ATMs and simple banking software manage user accounts, security, and transactions.

2. Objectives

- To simulate a real ATM's essential features such as registration, login, deposit, withdrawal, and PIN management.
- To implement secure PIN storage using cryptographic hashing.
- To create a persistent data storage mechanism using Python's pickle module.
- To reinforce fundamental programming concepts like functions, classes, and modularization.
- To provide a simple, user-friendly command-line interface.

3. File Descriptions

- atm.py: Handles all user interactions, displays menus, and manages the application flow.
- database.py: Manages account creation, authentication, and persistent storage of user data.
- user.py: Contains the User class, which encapsulates account operations like deposit, withdrawal, and PIN change.
- utils.py: Provides helper functions for hashing PINs, clearing the screen, and pausing the program.
- users.db: A binary file (auto-generated) used for storing user account information securely.
- README.md: Documentation for setup, usage, and project details.
- .gitignore: Specifies files and patterns to be ignored by version control.

4. Key Features

- User Registration: New users can create an account by providing their name and setting a 4-digit PIN.
- Authentication: Users log in using their unique account number and PIN, which is securely hashed.
- Balance Inquiry: Users can check their current balance at any time.
- Deposit: Users can deposit money into their account. The system ensures the deposit amount is at least 1.
- Withdrawal: Users can withdraw money, provided their account has sufficient funds.
- PIN Change: Users can securely update their PIN after confirming their old PIN.
- Account Details: Users can view account-specific information, such as their name, account number, and balance.
- Data Persistence: All account information is saved in a file (`users.db`), ensuring data remains after program restarts.
- Input Validation: The program validates all user inputs to prevent invalid operations and ensure data integrity.

5. Security Measures

- PIN Hashing: All PINs are hashed using SHA-256 before being stored, ensuring that actual PIN values are never saved in plain text.
- Input Checks: Deposits and withdrawals are validated for positive values, and withdrawals are limited to available balances.
- Account Authentication: Only users with correct account numbers and PINs can access their accounts.

6. How the Program Works

6.1. Startup: The user is presented with a main menu offering options to register, login, or exit.

6.2. <u>Registration</u>: Selecting registration prompts the user to enter their name and set a 4-digit PIN. Upon successful registration, the system generates a unique account number.

6.3. <u>Login</u>: Existing users provide their account number and PIN to access their account.

6.4. <u>ATM Operations</u>: After logging in, users can perform operations such as checking balance, depositing funds, withdrawing funds, changing their PIN, or viewing account details.

6.5. <u>Data Handling</u>: All changes (like deposits, withdrawals, PIN changes) are immediately written to the `users.db` file.

6.6. <u>Exit</u>: Users can logout or exit the application at any point, ensuring their data is safely stored.

7. Sample Code Snippet

Below is an example of the deposit logic, enforcing a minimum deposit amount:

```python
amt = input("Enter amount to deposit: ")
if amt.replace('.', '', 1).isdigit():
    amt = float(amt)
    if user.deposit(amt):
        db.update_user(user)
        print("Deposit successful.")
    else:
        print("Minimum deposit amount is 1.")
else:
    print("Invalid amount.")
```

8. Testing and Validation

The ATM interface was thoroughly tested for:

- <u>Registration & Account Creation</u>: Ensured new accounts are created with unique numbers, and PINs are securely stored.
- <u>Login</u>: Verified that only valid credentials provide access.
- <u>Deposits & Withdrawals</u>: Checked for correct balance updates and prevention of invalid transactions (such as negative or zero deposits and over-withdrawals).
- <u>PIN Change</u>: Confirmed that PIN changes require the correct old PIN and meet the format requirements.
- <u>Data Persistence</u>: Verified that all data persists between program runs.

9. Limitations and Future Enhancements

- <u>Limitations</u>:
    - No multi-user concurrency (single user at a time).
    - Lacks advanced security (no account lockout on repeated failed logins).
    - No transaction history or receipt printing.
    - File-based storage is not suitable for large-scale or high-security applications.

- <u>Possible Enhancements</u>:
    - Implement transaction history for each user.
    - Add account lockout or CAPTCHA after multiple failed logins.
    - Migrate to a database system (like SQLite) for better scalability and security.
    - Develop a graphical user interface (GUI) using Tkinter or PyQt.
    - Add email/SMS notifications for transactions.
    - Implement role-based access (e.g., admin features).

10. <u>Conclusion</u>

The Python ATM Interface project provides a comprehensive simulation of essential ATM operations within a secure and user-friendly command-line environment. It reinforces best practices in secure programming, user input validation, and modular code design. This project can serve as a foundation for more advanced banking or financial software and is an excellent starting point for students and beginners in Python programming.

11. <u>References</u>

- Python Official Documentation. https://docs.python.org/3
- ATM Machine in Python (GeeksforGeeks). https://www.geeksforgeeks.org/atm-machine-in-python

----