🏦 SMB BANK

A Documentary Small-to-Medium Bank (SMB) Ecosystem

🚀 Project Overview: The Digital Frontier

SMB Connect is a fully functional, simulated banking application designed to showcase a modern, secure, and user-friendly mobile experience for both clients and bank agents. Built on Python and KivyMD, this project provides a tangible demonstration of fundamental financial transaction logic, database management, and cross-platform UI development.
It is more than just a set of files; it is a **proof-of-concept for a complete digital banking infrastructure**, providing essential financial services through a single, unified interface.

Target Audience

1. **Clients:** End-users requiring seamless access to their checking, savings, and investment accounts.
2. **Bank Agents:** Staff members needing secure access for administrative tasks like account creation and password management.

✨ Core Features: Bridging Service and Convenience

SMB Connect implements a robust set of features categorized by user role.

Client Services (Accessible via main.py)

Feature	Description	Architectural Highlight
Full Transaction Suite	Deposit, Withdrawal, and Account-to-Account Transfer logic.	Atomic transaction handling within database.py
Real-Time Visibility	Check account and savings balances instantly.	Direct query integration with UI updates.
Comprehensive History	View a full log of all past transactions.	Dedicated transactions table with detailed metadata.
Secure Authentication	Client login and secure password management (SHA-256 hashing).	Hashing applied via _hash_password utility.
Dedicated Savings	Manage a separate savings account for goals and investments.	Implements a simulated **30% monthly interest rate** on savings.
Airtime Purchase	Deduct funds from the main account to purchase airtime for	Specialized transaction logging for target tracking.

Feature	Description	Architectural Highlight
	any phone number.	
Password Management	Clients can securely update their own passwords.	Uses dedicated screens for secure input.

Agent/Staff Utilities (Protected Access)

Feature	Description	Architectural Highlight
Agent Login	Secure access protected by a shared secret (STAFF_SECRET).	Separate staff_accounts table for privileged access control.
Account Lifecycle	Create new client accounts, view all accounts, and delete inactive accounts.	Centralized CRUD operations controlled by agent logic.
Client Search	Verify and retrieve specific client details for service assistance.	Efficient database querying by account number.
Password Reset	Agent-initiated password reset capability for clients.	Essential administrative feature for customer support.

🏗️ Architecture & Technology Stack

The application adheres to a clean separation of concerns, ensuring maintainability and scalability, even in a single-file KivyMD structure.

Layer	Files Involved	Technology	Responsibility
Presentation (UI)	main.py	Python, KivyMD, Kivy KV Language	Screen management, user input handling, visual design, and event triggering.
Business Logic	database.py	Python	All financial validation (e.g., sufficient funds, interest calculation) and core functional execution.
Persistence	database.py (via internal methods)	SQLite 3	Data storage for accounts, transactions, and staff data. Guarantees data integrity.

📁 File Manifest

File	Role	Key Functions
main.py	**The Frontend Interface**	Contains all KivyMD screen definitions (KV language) and the application core (SmbApp). Manages state and calls database.py functions based on user interaction.

File	Role	Key Functions
database.py	**The Backend Engine**	Handles all interactions with the SQLite database. Contains critical functions for authentication, transactions, account management, and the business logic for interest and airtime purchases.

⚙️ Setup and Execution

Prerequisites

1. **Python 3.x**
2. **Kivy** (pip install kivy)
3. **KivyMD** (pip install kivymd)

Getting Started

1. **Ensure File Integrity:** Verify that both main.py and database.py are in the same working directory.
2. **Database Initialization:** The main.py script automatically calls database.setup() on startup. This initializes the SQLite database (smb_bank.db), creates all necessary tables (accounts, transactions, staff_accounts), and inserts a default staff account for agent login.
3. **Run the Application:**
   ```
   python main.py
   ```

Important Notes

- **Default Staff Secret:** The initial agent login uses the secret defined in database.py (STAFF_SECRET = 'smb_staff_pass'). This provides immediate access to administrative features.
- **Security:** Passwords are securely hashed using SHA-256 before storage.
- **Testing:** We recommend creating a test account through the Agent screen to fully explore the Client features (Deposit, Transfer, Airtime, Savings).

💡 Future Direction

This project serves as a strong foundation. Potential expansions could include:

- **Security Enhancements:** Implementing MFA/2FA simulation.
- **Advanced Features:** Adding loan processing, budget tracking, or recurring payment schedules.
- **Deployment:** Converting the application to a native mobile package (APK/iOS app).
- **External Integration:** Replacing SQLite with a cloud-hosted database (e.g., Firestore) for multi-user, distributed access.