TECHNICAL REPORT

Problem Statement

With the increasing use of digital banking systems, ensuring the security of user data and financial transactions has become a critical challenge. Traditional systems are often vulnerable to data breaches, unauthorized access, and insecure storage of sensitive information such as passwords and account balances.

This project addresses the problem of **secure user authentication, protected data storage, and safe transaction handling** in an online banking environment.

Dataset

The dataset in this project is **application-generated data**, not a public dataset. It consists of:

- User credentials (usernames and hashed passwords)
- User account balances
- Transaction records (sender, receiver, amount, timestamp)

All sensitive data (balances and transactions) are **encrypted before storage** and saved in structured tables within a **Supabase cloud database**. This dataset simulates real banking data while maintaining a strong focus on confidentiality and integrity.

Method / System Design

The system is implemented using a **security-first architecture**:

- **Streamlit** is used to build the user interface.
- **Supabase** acts as the backend database service.
- **AES encryption (CBC mode)** is applied to balances and transaction details.
- **Bcrypt hashing** secures user passwords.
- **JWT (JSON Web Tokens)** manage user sessions and authentication.

The workflow includes secure user registration, login verification, encrypted balance retrieval, validated money transfers, and secure logout with session expiration.

Results

The system was successfully tested and produced the following results:

- Secure authentication using hashed passwords and JWT tokens.
- All sensitive financial data stored in encrypted form.
- Correct handling of transactions with balance validation.
- Prevention of unauthorized access, invalid transfers, and session misuse.

Since this is a security-based system rather than a machine learning model, **accuracy is measured by correct functionality and security enforcement**, which was achieved consistently.

Conclusion

This project demonstrates that combining encryption, secure authentication, and proper session management can result in a **robust and reliable banking application**. The system effectively protects sensitive data and ensures secure user interactions.

Future improvements may include audit logging, role-based access control, and advanced fraud detection mechanisms to further enhance system security.

Code Structure

```
SecureBankApp/
│
├── app/                          #  Main application package
│   ├── __init__.py
│   ├── main.py                   # Streamlit entry point (was SecureBankApp.py)
│   │
│   ├── agents/
│   │   ├── __init__.py
│   │   └── ai_agent.py
│   │
│   ├── database/
│   │   ├── __init__.py
│   │   ├── init_db.py
│   │   └── bank.db
│   │
│   ├── services/                 # (future: auth, transactions, logging)
│   │   └── __init__.py
│   │
│   ├── ui/
│   │   └── __init__.py           # UI logic if app grows
│   │
│   └── config/
│       └── settings.py
│
├── .streamlit/
│   └── secrets.toml              #  Streamlit secrets
│
├── docs/                         #  Documentation isolated
│   ├── Documentation.pdf
│   ├── README.md
│   └── Users_initialized.txt
│
├── requirements.txt
├── .gitignore
└── README.md                     # Project overview (root-level)
```