Mini Banking Application Using Procedural Programming

Objective

Implement a simple banking system using **procedural programming** in Python. The system should allow users to create accounts, deposit money, withdraw money, check balances, and view transaction history.

Requirements

1. Account Creation

- The system should allow users to create a new bank account with:
    - **Account number** (unique identifier - Auto-Generated Account Numbers)
    - **Account holder name** ○
    - **Initial balance** (must be non-negative)
- If an account number already exists, the system should display an error message.

2. Deposit Money

- Users should be able to deposit money into an existing account.
- The deposit amount must be a **positive number**.
- The system should update the account balance and record the transaction.

3. Withdraw Money

- Users should be able to withdraw money from an existing account.
- The withdrawal amount must be a **positive number** and **should not exceed the available balance**.
- The system should update the account balance and record the transaction.

4. Check Balance

- Users should be able to check the current balance of an account.

5. Transaction History

- Users should be able to view all past transactions (deposits and withdrawals) for a given account.

6. Menu-Driven Interface

- The system should present a **menu** with the following options:

  1. **Create Account** 2. **Deposit Money** 3. **Withdraw Money** 4. **Check Balance** 5. **Transaction History** 6. **Exit**

- The program should run until the user chooses to exit.

Implementation Guidelines

1. Use **dictionaries** to store account details (account number as the key).
2. Each account should store: ○ Account holder name ○ Current balance ○ List of transactions (deposits and withdrawals)
3. Use **functions** for each operation (e.g., create_account(), deposit_money(), etc.).
4. Handle **input validation** (e.g., negative amounts, invalid account numbers).
5. Ensure the program runs in a loop until the user chooses to exit.

Bonus (Optional)

1. Add a feature to **transfer money** between two accounts.
2. Implement **password protection** for accounts.
3. Save account data to a **file** (for persistent storage).
4. Interest Calculation

Submission Instructions

- Submit a **single Python file** (banking_app.py) with your implementation.
- Include **comments** explaining key parts of your code.
- Ensure your program runs without errors.

Evaluation Criteria

1. **Functionality** (All required features work correctly)
2. **Code Structure** (Proper use of functions, variables, and data structures)
3. **Input Validation** (Handles invalid inputs gracefully)
4. **Error Handling** (Displays appropriate error messages)