```
import json
import random
from datetime import datetime, timedelta
import os

class BankingChatbot:
    def __init__(self):
        self.data_file = "bank_data.json"
        self.accounts = {}
        self.current_account = None
        self.load_data()

        # Initialize with sample data if file doesn't exist
        if not self.accounts:
            self.initialize_sample_data()
            self.save_data()

    def load_data(self):
        """Load account data from JSON file"""
        if os.path.exists(self.data_file):
            with open(self.data_file, 'r') as f:
                self.accounts = json.load(f)

    def save_data(self):
        """Save account data to JSON file"""
        with open(self.data_file, 'w') as f:
            json.dump(self.accounts, f, indent=2)

    def initialize_sample_data(self):
        """Generate sample data for 1000 accounts"""
        first_names = ["James", "Mary", "John", "Patricia", "Robert", "Jennifer",
                       "Michael", "Linda", "William", "Elizabeth", "David", "Barbara",
                       "Richard", "Susan", "Joseph", "Jessica", "Thomas", "Sarah"]
        last_names = ["Smith", "Johnson", "Williams", "Brown", "Jones", "Miller",
                      "Davis", "Garcia", "Rodriguez", "Wilson", "Martinez", "Anderson"]

        for i in range(1000):
            account_num = str(100000 + i)
            first = random.choice(first_names)
            last = random.choice(last_names)
            pin = f"{random.randint(0,9999):04d}"
```

```
        balance = round(random.uniform(100, 50000), 2)

        # Generate transaction history (3-20 transactions per account)
        transactions = []
        tx_count = random.randint(3, 20)
        for _ in range(tx_count):
            days_ago = random.randint(1, 365)
            tx_date = (datetime.now() - timedelta(days=days_ago)).strftime("%Y-%m-%d")

            tx_types = ["Deposit", "Withdrawal", "Transfer", "Salary", "Payment", "Purchase"]
            tx_type = random.choice(tx_types)

            if tx_type in ["Deposit", "Salary", "Transfer"]:
                amount = round(random.uniform(50, 5000), 2)
            else:
                amount = round(random.uniform(-5000, -5), 2)

            if tx_type == "Transfer":
                target_account = str(100000 + random.randint(0, 999))
                description = f"{tx_type} to {target_account}"
            else:
                description = tx_type

            transactions.append({
                "date": tx_date,
                "description": description,
                "amount": amount
            })

        self.accounts[account_num] = {
            "pin": pin,
            "balance": balance,
            "name": f"{first} {last}",
            "transactions": transactions,
            "account_type": random.choice(["Checking", "Savings"]),
            "email": f"{first.lower()}.{last.lower()}@example.com",
            "phone": f"({random.randint(200,999)}) {random.randint(200,999)}-
{random.randint(1000,9999)}"
        }

    def authenticate(self, account_number, pin):
```

```
        """Authenticate user with account number and PIN"""
        if account_number in self.accounts and self.accounts[account_number]["pin"] ==
pin:
            self.current_account = account_number
            return True, f"Welcome back, {self.accounts[account_number]['name']}!"
        return False, "Invalid account number or PIN. Please try again."

    def get_account_info(self):
        """Return comprehensive account information"""
        if self.current_account:
            acc = self.accounts[self.current_account]
            info = f"""
            Account Holder: {acc['name']}
            Account Number: {self.current_account}
            Account Type: {acc['account_type']}
            Current Balance: ${acc['balance']:.2f}
            Email: {acc['email']}
            Phone: {acc['phone']}
            """
            return info
        return "Please log in to view account information."

    def get_balance(self):
        """Return current account balance"""
        if self.current_account:
            return f"Your current balance is
${self.accounts[self.current_account]['balance']:.2f}"
        return "Please log in to check your balance."

    def get_transaction_history(self, limit=10):
        """Return transaction history with optional limit"""
        if self.current_account:
            transactions = sorted(
                self.accounts[self.current_account]["transactions"],
                key=lambda x: x["date"],
                reverse=True
            )[:limit]

            history = f"Your recent transactions (last {len(transitions)}):\n"
            for tx in transactions:
                history += f"{tx['date']}: {tx['description']} - ${tx['amount']:.2f}\n"
```

```
        return history
    return "Please log in to view your transaction history."

def transfer_funds(self, target_account, amount):
    """Transfer funds to another account"""
    if not self.current_account:
        return "Please log in to transfer funds."

    if self.current_account == target_account:
        return "Cannot transfer to the same account."

    if target_account not in self.accounts:
        return "Recipient account not found."

    if amount <= 0:
        return "Amount must be positive."

    if self.accounts[self.current_account]["balance"] < amount:
        return "Insufficient funds for this transfer."

    # Perform the transfer
    self.accounts[self.current_account]["balance"] -= amount
    self.accounts[target_account]["balance"] += amount

    # Record transactions for both accounts
    today = datetime.now().strftime("%Y-%m-%d")
    self.accounts[self.current_account]["transactions"].append({
        "date": today,
        "description": f"Transfer to {target_account}",
        "amount": -amount
    })

    self.accounts[target_account]["transactions"].append({
        "date": today,
        "description": f"Transfer from {self.current_account}",
        "amount": amount
    })

    self.save_data()
    return f"Successfully transferred ${amount:.2f} to account {target_account}."
```

```
    def change_pin(self, new_pin):
        """Change account PIN"""
        if self.current_account:
            if len(new_pin) == 4 and new_pin.isdigit():
                self.accounts[self.current_account]["pin"] = new_pin
                self.save_data()
                return "Your PIN has been successfully changed."
            return "PIN must be 4 digits."
        return "Please log in to change your PIN."

    def logout(self):
        """Log out of current account"""
        if self.current_account:
            account_name = self.accounts[self.current_account]["name"]
            self.current_account = None
            return f"Goodbye, {account_name}! You have been logged out."
        return "No account is currently logged in."


def main():
    chatbot = BankingChatbot()
    print("Welcome to Advanced BankBot! How can I assist you today?")

    while True:
        print("\nMain Menu:")
        print("1. Login")
        print("2. Account Information")
        print("3. Check Balance")
        print("4. View Transactions")
        print("5. Transfer Money")
        print("6. Change PIN")
        print("7. Logout")
        print("8. Exit")

        choice = input("Enter your choice (1-8): ")

        if choice == "1":  # Login
            account_number = input("Enter your account number: ")
            pin = input("Enter your PIN: ")
            success, message = chatbot.authenticate(account_number, pin)
            print(message)
```

```
elif choice == "2":  # Account Information
    print(chatbot.get_account_info())

elif choice == "3":  # Check Balance
    print(chatbot.get_balance())

elif choice == "4":  # View Transactions
    limit = input("How many transactions to show? (default 10): ")
    try:
        limit = int(limit) if limit else 10
        print(chatbot.get_transaction_history(limit))
    except ValueError:
        print("Invalid number. Showing default 10 transactions.")
        print(chatbot.get_transaction_history())

elif choice == "5":  # Transfer Money
    if not chatbot.current_account:
        print("Please log in first.")
        continue
    target_account = input("Enter recipient account number: ")
    try:
        amount = float(input("Enter amount to transfer: "))
        print(chatbot.transfer_funds(target_account, amount))
    except ValueError:
        print("Invalid amount. Please enter a number.")

elif choice == "6":  # Change PIN
    new_pin = input("Enter your new 4-digit PIN: ")
    print(chatbot.change_pin(new_pin))

elif choice == "7":  # Logout
    print(chatbot.logout())

elif choice == "8":  # Exit
    if chatbot.current_account:
        print(chatbot.logout())
    print("Thank you for using Advanced BankBot. Have a great day!")
    break

else:
```

```
        print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```