Python Bank Account Management System

Objective

Create a Python program that simulates a bank account management system. The system should allow users to create and manage bank accounts, perform transactions, and retrieve relevant details. The project should incorporate key Python concepts, such as variables, data structures, functions, file handling, and modules.

Requirements

1. Account Management

- Users should be able to create new bank accounts. Each account must have the following attributes:
  - Account holder's name
  - Account number (auto-generated)
  - Account type (e.g., savings or current)
  - Initial balance
- Allow users to retrieve account details, including:
  - Account holder's name
  - Account number
  - Account type
  - Current balance

2. Transactions

- Implement the following transactions:
  - **Deposit:** Users can deposit money into their accounts.
  - **Withdrawal:** Users can withdraw money, ensuring that the withdrawal does not exceed the current balance.
  - **Transfer:** Implement a function to transfer funds between two accounts.
- Ensure that transactions update the account balance accordingly.

3. File Handling

- Store account details and transaction history in files.
- Implement functionality to read and write account data to a file (using the `pickle` library).
- Load account data from the file at the start of the program and save updates to the file when the program terminates.
- Ensure the program can append new transactions to the existing file without overwriting previous data.

4. Reports

- Allow users to view transaction history for a specific account.
- Show details like:
  - Date
  - Type of transaction (deposit, withdrawal, transfer)
  - Amount
- Generate summary statistics for each account, such as:
  - Total deposits
  - Total withdrawals
  - Average transaction amounts (using `NumPy` functions)

5. User Interaction

- Create an interactive menu to perform the following operations:
  - Open a new account
  - View account details
  - Perform transactions (deposit, withdraw, transfer)
  - View transaction history
  - Exit the program
- Use conditional statements and loops to ensure smooth navigation between different menu options.

6. Error Handling

- Implement input validation (e.g., ensure deposit and withdrawal amounts are positive).

- Handle potential errors, such as:
  - Attempting to withdraw more money than available in the account.
  - Transferring funds between non-existent accounts.

7. Bonus (Optional)

- Implement login functionality for multiple users, requiring a username and password to access each account.

- Use advanced Python features, such as lambda functions, for quick calculations or specific operations within the program.