**Banking System**  
  
**Overview**This project is a simple banking system that allows the creation and management of customers, accounts, saving and current, and transactions. The system provides functionalities to create customers and accounts, perform transactions, check account balance, and retrieve the transaction history.  
  
**File**- `saving\_accounts.txt`: Saves data on saving accounts.  
- `current\_accounts.txt`: Saves data on current accounts.  
- `customers.txt`: Saves data on customers.  
- `transactions.txt`: Saves data on transactions.  
  
**Classes**  
**Account**Represents a generic bank account.  
  
**Attributes**- `account\_number`: Unique identifier for the account.  
- `balance`: Current balance in the account.  
- `customer`: The customer who owns the account.  
  
**Methods**- `deposit(amount)`: Adds the specified amount to the account balance.  
- `withdraw(amount)`: Deducts the specified amount from the account balance if sufficient funds are available.  
- `check\_balance()`: Returns the current balance of the account.  
  
**SavingAccount**Represents a saving account; inherits from Account.  
  
**Attributes**- Inherits all attributes from Account.  
- `interest\_rate`: A class attribute, which represents the interest rate; default is 0.02.  
  
**Methods**- `add\_monthly\_interest()`: Adds monthly interest to the account balance.  
  
**CurrentAccount**Represents a current account; inherits from Account.  
  
**Attributes**- Inherits all attributes from Account.  
- `overdraw\_limit`: Maximum amount that can be overdrawn from the account.  
  
**Customer**Represents a customer.  
  
**Attributes**  
- `name`: Name of the customer.  
- `address`: Address of the customer.  
- `contact\_details`: Contact details of the customer.  
- `accounts`: List of accounts owned by the customer.  
  
**Methods**  
- `add\_account(account)`: Adds an account to the customer's list of accounts.  
  
**Transaction**  
Represents a transaction made on an account.  
  
**Attributes**  
- `transaction\_id`: Unique identifier for the transaction.  
- `timestamp`: The time when the transaction was made.  
- `account`: The account involved in the transaction.  
- `transaction\_type`: The type of transaction; e.g. deposit or withdrawal.  
- `amount`: The amount involved in the transaction.  
  
**Functions**

**load\_data()**Loads data from the files and returns it as dictionaries and lists.  
  
**save\_data(saving\_accounts\_data, current\_accounts\_data, customers\_data, transactions\_data)**Saves the provided data to the corresponding files.  
  
**create\_customer()**Asks the user for customer details and creates a new Customer object.  
  
**create\_account(customer)**Asks the user for account details and creates a new Account, either SavingAccount or CurrentAccount, for the given customer.  
  
**make\_transaction(account)**Asks the user for transaction details and performs the transaction on the given account.  
  
**check\_balance(account)**Displays the current balance of the given account.  
  
**view\_transaction\_history(transactions\_data)**Shows the transaction history from transactions data provided.  
  
**main()**The main function runs the command-line interface for the banking system.

**Usage**

**Create Customer**  
- Customers enter their name, address, and contact information.  
- The customer is then stored within the customers.txt file.  
  
**Create Account**- Customers enter their name.  
- Select the type of account: Saving or Current.  
- Enter account number and initial account balance, and the overdraw limit (if any).  
- The account is then stored in its corresponding account file (`saving\_accounts.txt` or `current\_accounts.txt`).  
  
**Make Transaction**- Customer enters the account number.  
- Select transaction type: Deposit or Withdrawal.  
- Enter the amount of the transaction.  
- The transaction is then written into the `transactions.txt` file.  
  
**Check Balance**- Customer enters the account number.  
- The account balance is then shown.  
  
**View Transaction History**- Shows all transactions written into the `transactions.txt` file.  
  
**Exit**- Data in the files is then stored and the program will exit.  
  
**Example**Here is an example of how the system can be used:  
- A customer is created whose name is John Doe.  
- A saving account is opened for John Doe with an initial balance of $1000.  
- A deposit of $200 is made to the saving account of John Doe.  
- The current balance of John Doe's saving account is checked.  
- Transaction history is viewed.  
  
**Conclusion**This is a simple banking system in which customers, their accounts, and transactions are provided. It can be extended and modified to include additional features and functionality according to needs.