Bank Account System:

1. Have Used Dictionary to store the account number as key, values as name and initial\_amount.
2. We have Two Types of bank Accounts the first one is Savings account, and the second one is Checking account.
3. Saving Account is used here for saving the money with 4% interest rate annually.
4. Checking bank account is also called as current account, we can deposit and withdraw money frequently.
5. We have an overdraft limit (maximum amount that can be withdrawn beyond the account balance) for Checking account. In our case 500.
6. We have a Log list where the transaction details are been stored.
7. We also have implemented Custom Error Handling in deposit, withdraw, and our menu.
8. We have used Decorators to automatically log every deposit or withdrawal instead of manually writing self.\_transactions.append(). Inside each method.
9. Have used Pytest to write simple unit tests and automated test cases.

We have 4 Classes

1.BankAccount (Base Class).

2.SavingAccount: this will inherit the properties of BankAccount.

3.CheckingAccount: this will inherit the properties of BankAccount.

4.Transactions: Stores all type of transactions (deposit, withdrawal, account creation, amount, date and time and optional message)

In BankAccount we have a constructer and 4 methods (deposit, withdraw, see\_balance and display\_details).

In constructor we have name, Initial \_amount and initial\_counter with incrementation.

IN CheckingAccount Class:

== In CheckingBank class we have a concept called Overdraft:

Overdraft means we can allow the user to withdraw money even if they don’t have the required balance amount. EX. We have 100rs in our account and the overdraft limit is set to 500 and we want to withdraw 400rs, now we can withdraw 400rs

We will override the withdraw method from the base class

IN Transaction Class:

We have three parameters, those are type of transaction (typeot), amount, note.

We create a constructor and while declaring the variables we declare variables for typeoft, amount and note we also create one more variable called timestamp and give the value as datetime.now (), and we create a one more method called as \_\_str\_\_ which return the timestamp, amount and the note.

Changes in main class are:

We create a private list called as transactions, and the. append is used to store the log operation, like this it has been appended when we execute deposit and withdraw methods.

Then we create one more function called as show\_transactions and display the list using for loop.

Creating a Class for Custom error handling:

First, we create a class called BankError and give Exception as parameter, and we create our own custom error handling classes.

So we create 3 custom classes first one is InvalidAmountError this comes when we enter a zero or negative amount.

Second one is InsufficientError this comes when there is no amount in the account.

Third one is AccountNotFoundError this comes when there is no account found.

+ We implement all this in withdraw and deposit method.

We Create a Decorator:

A decorator is a function that wraps another function to add extra behaviour without changing the original function’s code.

We should put the decorator above the BankAccount class.

We will create a decorator that adds a transaction log record automatically whenever a method like deposit() or withdraw() is called.

We have used generators in our project because no need to hold all transactions in the memory and if we are building an app it will slow it or crash ,and it is scalable also ,mainly used for large datasets, it is lazy loading.