BANK MANAGEMENT SYSTEM

Description Overview:

The Bank Management System implemented in the provided code offers basic functionalities for managing banking operations.

Here's an overview of its key features:

User Management: Users can log in using their credentials. New users can also be created with unique usernames and passwords.

Account Management: The system allows the creation of bank accounts associated with account numbers and holder names. Initial balances for accounts can be specified during creation.

Credit Card Management: Users can create credit cards linked to their bank accounts. Credit limits for these cards can be set during creation.

Loan Management: The system supports the creation of loans for bank accounts. Users can specify the loan amount, and the system calculates GST (Goods and Services Tax) accordingly.

Deposit and Withdrawal: Users can deposit funds into their accounts and withdraw funds as needed. The system ensures that withdrawals do not exceed the available balance.

Balance Inquiry: Users can check the balance of their accounts at any time

Problem Statement:

Create a Bank Management System that allows users to login, create accounts, credit cards, loans, deposit, withdraw, and check balance. The system should provide a user-friendly interface for managing banking operations.

Technology Used:

- Here we have used google collab which was introduced by google research team and python programming language where it’s easy to code & learn

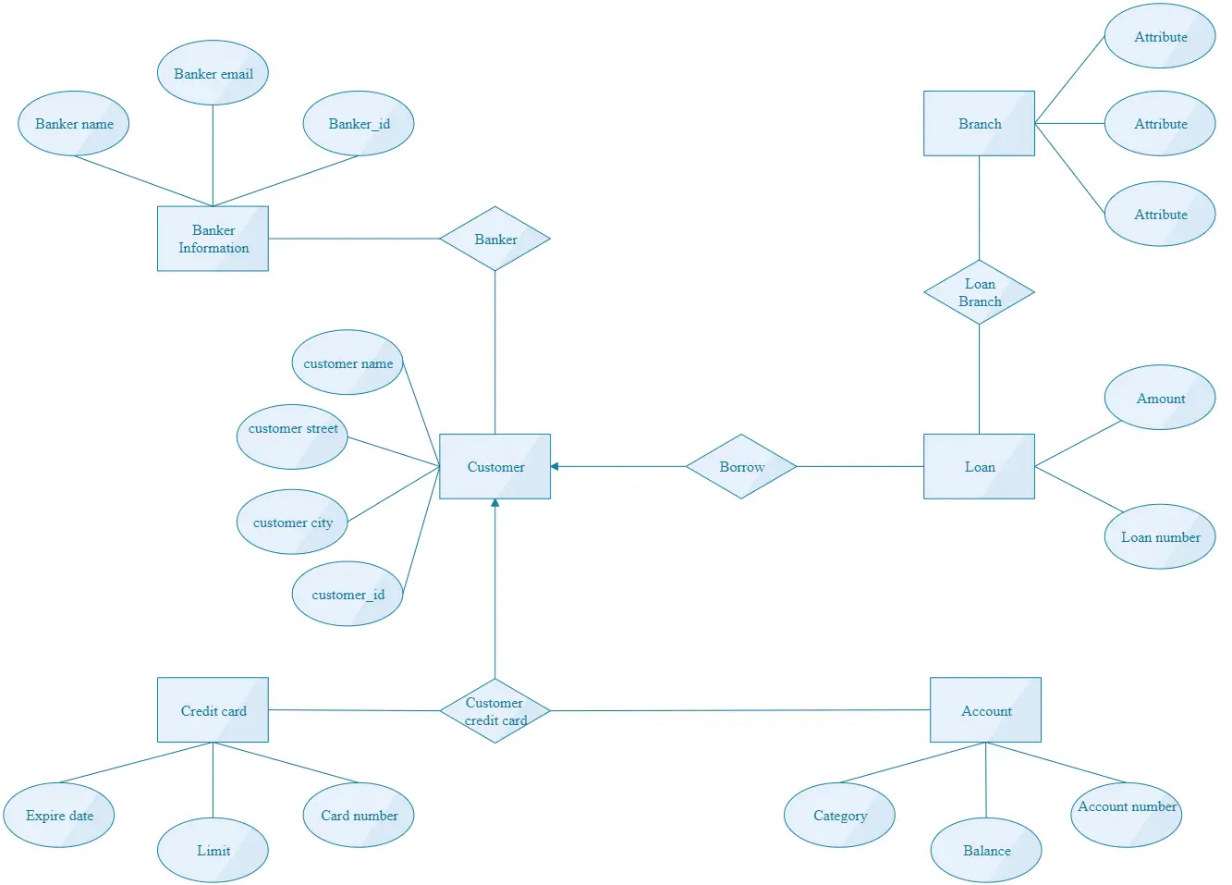
Installation:

1. Ensure Python 3.x is installed on your system.

2. Download the provided Python script.

3. Run the script in your preferred Python environment.

Workflow of Code:



Implementation:

class Bank:

def \_\_init\_\_(self):

self.accounts = {}

self.credit\_cards = {}

self.loans = {}

self.users = {}

def login(self, username, password):

return username in self.users and self.users[username] == password

def create\_user(self, username, password):

if username not in self.users:

self.users[username] = password

print(f"User {username} created successfully!")

else:

print("Username already exists. Please choose another one.")

def create\_account(self, account\_number, holder\_name, initial\_balance=0):

if account\_number not in self.accounts:

self.accounts[account\_number] = {"holder\_name": holder\_name, "balance": initial\_balance}

print(f"Account created for {holder\_name} with account number {account\_number}")

else:

print(f"Account with account number {account\_number} already exists.")

def create\_credit\_card(self, account\_number, credit\_limit=0):

if account\_number in self.accounts and account\_number not in self.credit\_cards:

self.credit\_cards[account\_number] = {"credit\_limit": credit\_limit, "balance": 0}

print(f"Credit card created for account number {account\_number} with credit limit {credit\_limit}")

elif account\_number not in self.accounts:

print(f"Account with account number {account\_number} not found.")

else:

print(f"Credit card already exists for account number {account\_number}.")

def create\_loan(self, account\_number, loan\_amount=0):

if account\_number in self.accounts and account\_number not in self.loans:

gst = loan\_amount \* 0.3

total\_amount = loan\_amount + gst

self.loans[account\_number] = {"loan\_amount": loan\_amount, "gst": gst, "total\_amount": total\_amount,

"remaining\_amount": total\_amount} # Initialize remaining\_amount

print(f"Loan created for account number {account\_number} with loan amount {loan\_amount} and GST {gst}. Total amount: {total\_amount}")

elif account\_number not in self.accounts:

print(f"Account with account number {account\_number} not found.")

else:

print(f"Loan already exists for account number {account\_number}.")

def pay\_loan(self, account\_number, amount):

if account\_number in self.accounts and account\_number in self.loans:

if self.loans[account\_number]["remaining\_amount"] >= amount:

self.loans[account\_number]["remaining\_amount"] -= amount

print(

f"Paid {amount} towards the loan for account {account\_number}. Remaining amount: {self.loans[account\_number]['remaining\_amount']}")

else:

print(f"Exceeded remaining loan amount. Cannot pay more than the remaining loan amount.")

elif account\_number not in self.accounts:

print(f"Account with account number {account\_number} not found.")

else:

print(f"No loan found for account number {account\_number}.")

def deposit(self, account\_number, amount):

if account\_number in self.accounts:

self.accounts[account\_number]["balance"] += amount

print(

f"Deposited {amount} into account {account\_number}. New balance: {self.accounts[account\_number]['balance']}")

else:

print(f"Account with account number {account\_number} not found.")

def withdraw(self, account\_number, amount):

if account\_number in self.accounts:

if self.accounts[account\_number]["balance"] >= amount:

self.accounts[account\_number]["balance"] -= amount

print(

f"Withdrew {amount} from account {account\_number}. New balance: {self.accounts[account\_number]['balance']}")

else:

print(f"Insufficient funds in account {account\_number}.")

else:

print(f"Account with account number {account\_number} not found.")

def check\_balance(self, account\_number):

if account\_number in self.accounts:

balance = self.accounts[account\_number]["balance"]

print(f"Balance in account {account\_number}: {balance}")

else:

print(f"Account with account number {account\_number} not found.")

def main():

bank = Bank()

while True:

print("\nBank Management System:")

print("1. Login")

print("2. Create User")

print("3. Create Account")

print("4. Create Credit Card")

print("5. Create Loan")

print("6. Pay Loan")

print("7. Deposit")

print("8. Withdraw")

print("9. Check Balance")

print("10. Exit")

choice = input("Enter your choice (1-10): ")

if choice == "1":

username = input("Enter username: ")

password = input("Enter password: ")

if bank.login(username, password):

print("Login successful!")

else:

print("Invalid username or password.")

elif choice == "2":

new\_username = input("Enter new username: ")

new\_password = input("Enter new password: ")

bank.create\_user(new\_username, new\_password)

elif choice == "3":

account\_number = input("Enter account number: ")

holder\_name = input("Enter account holder name: ")

initial\_balance = float(input("Enter initial balance (default is 0): ") or 0)

bank.create\_account(account\_number, holder\_name, initial\_balance)

elif choice == "4":

account\_number = input("Enter account number: ")

credit\_limit = float(input("Enter credit limit: "))

bank.create\_credit\_card(account\_number, credit\_limit)

elif choice == "5":

account\_number = input("Enter account number: ")

loan\_amount = float(input("Enter loan amount: "))

bank.create\_loan(account\_number, loan\_amount)

elif choice == "6":

account\_number = input("Enter account number: ")

amount = float(input("Enter payment amount towards the loan: "))

bank.pay\_loan(account\_number, amount)

elif choice == "7":

account\_number = input("Enter account number: ")

amount = float(input("Enter deposit amount: "))

bank.deposit(account\_number, amount)

elif choice == "8":

account\_number = input("Enter account number: ")

amount = float(input("Enter withdrawal amount: "))

bank.withdraw(account\_number, amount)

elif choice == "9":

account\_number = input("Enter account number: ")

bank.check\_balance(account\_number)

elif choice == "10":

print("Exiting the Bank Management System. Goodbye!")

break

else:

print("Invalid choice. Please enter a number between 1 and 10.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

Testing in Local/API:

Testing can be performed by running the script locally and interacting with the menu interface. API testing can be implemented by exposing the functionalities as API endpoints and sending requests to these endpoints.

Conclusion:

The Bank Management System provides basic functionalities for managing banking operations. It can be extended further to incorporate additional features such as transaction history, interest calculations, and account statements.

Comparison:

This script provides a simple implementation of a Bank Management System. More advanced systems may include features like multi-factor authentication, encryption for sensitive data, integration with external services for transactions, and comprehensive reporting capabilities.