from pymongo import MongoClient

from flask\_bcrypt import Bcrypt

client = MongoClient('mongodb://localhost:1999/')

db = client.bank

bcrypt = Bcrypt()

class User:

def \_\_init\_\_(self, username, password, role='customer'):

self.username = username

self.password = bcrypt.generate\_password\_hash(password).decode('utf-8')

self.role = role

class Account:

def \_\_init\_\_(self, account\_number, owner, balance=0.0):

self.account\_number = account\_number

self.owner = owner

self.balance = balance

class Bank:

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

self.db = db

self.accounts = db.accounts

self.users = db.users

self.transactions = db.transactions

def create\_user(self, username, password, role='customer'):

if self.users.find\_one({"username": username}) is None:

user = User(username, password, role)

self.users.insert\_one(user.\_\_dict\_\_)

return True

return False

def create\_account(self, account\_number, owner):

if self.accounts.find\_one({"account\_number": account\_number}) is None:

account = Account(account\_number, owner)

self.accounts.insert\_one(account.\_\_dict\_\_)

return True

return False

def get\_account(self, account\_number):

account\_data = self.accounts.find\_one({"account\_number": account\_number})

if account\_data:

return Account(\*\*account\_data)

return None

def update\_account(self, account):

self.accounts.update\_one(

{"account\_number": account.account\_number},

{"$set": {"balance": account.balance}}

)

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

account = self.get\_account(account\_number)

if account and amount > 0:

account.balance += amount

self.update\_account(account)

self.transactions.insert\_one({"account\_number": account\_number, "type": "deposit", "amount": amount})

return True

return False

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

account = self.get\_account(account\_number)

if account and 0 < amount <= account.balance:

account.balance -= amount

self.update\_account(account)

self.transactions.insert\_one({"account\_number": account\_number, "type": "withdrawal", "amount": amount})

return True

return False

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

account = self.get\_account(account\_number)

if account:

return account.balance

return None

def get\_transaction\_history(self, account\_number):

return list(self.transactions.find({"account\_number": account\_number}))

def get\_all\_accounts(self):

return list(self.accounts.find())