NJALA UNIVERSITY

School of Technology

Department of Computer Science & Information Technology

Project Title:

Designing and Implementation of an Online Banking Application

Submitted by:

(1) Tommy Daniel Tucker: ID.: 19545

(2) Abu Junior Vandi: ID.:82937

(3) Joshua Joe Harding: ID.: 19418

(4) Hawanatu Kabba Bangura: ID.: 55072

Name of Tutor.: Dr. AJ. Fofanah

Courses:
CSP610-Object Oriented Programming &CSP611-Advanced data Structure & Algorithm

Submission Date:
30/08/2024

1. Abstract

1.1 Background

The rapid advancements in technology and the widespread adoption of the internet have revolutionized the banking sector. Traditional banking methods that require physical presence at a bank branch are increasingly being replaced by digital banking systems. These systems provide customers with the convenience of managing their finances from anywhere at any time. This shift not only enhances user experience but also reduces operational costs for banks.

1.2 Problem Statement

Despite the availability of online banking solutions, many existing systems lack comprehensive features and user-friendly interfaces, leading to a suboptimal user experience. Furthermore, security concerns remain a critical challenge, with increasing incidents of cyber threats and fraudulent activities. There is a pressing need for a robust, secure, and efficient online banking system that addresses these issues and meets the modern demands of customers, providing a seamless and safe banking experience.

1.4 Main Features:

- **User Registration (Sign up):** Allows new users to create an account with secure credentials.
- **Login:** Enables existing users to access their accounts using their username and password/PIN.
- **Forget Pin:** Provides a mechanism for users to recover or reset their forgotten PIN.
- **Deposit Money:** Allows users to add funds to their bank account.
- **Withdraw Money:** Enables users to withdraw funds from their bank account.
- **Transfer Money:** Facilitates the transfer of funds between accounts.
- **Check Balance Inquiries:** Provides users with real-time access to their account balance and transaction history.
- **Deposit Interest Rate Calculation:** Calculates interest earned on deposits.
- **Compound Interest Calculation:** Computes compound interest for user deposits.
- **Exit Menu:** Allows users to securely log out of the system.

2. Objectives

2.1 Project Objectives

2.2 Aim

The aim of the Online Banking System project is to develop a secure, efficient, and user-friendly platform that allows users to manage their bank accounts and perform various financial transactions online. The system seeks to provide a comprehensive solution for modern banking needs, enhancing customer experience by offering convenience and accessibility, while ensuring data security and transaction integrity.

2.3 Scope of the Project

This project focuses on creating a software application that demonstrates the principles of Object-Oriented Programming (OOP) and the implementation of various data structures and algorithms. The core functionalities of the Online Banking System include:

- User Registration (Sign up)
- Secure Login
- PIN Recovery (Forget Pin)
- Deposit Money
- Withdraw Money
- Transfer Money
- Balance Inquiry
- Deposit Interest Rate Calculation
- Compound Interest Calculation
- Secure Logout (Exit Menu)

The system will be developed using Python as the primary programming language, leveraging the Math library for necessary calculations. The development environment will be Visual Studio Code, providing an efficient and feature-rich platform for coding and debugging.

2.4 Significance of the Study

This project aims to provide a comprehensive solution for modern banking needs by integrating essential banking functionalities into a single platform. By ensuring secure transactions and offering a user-friendly interface, the system will enhance customer satisfaction and streamline banking operations. Furthermore, the project serves as a practical implementation of theoretical concepts in computer science, particularly in OOP and advanced data structures and algorithms.

3. Technology

To successfully develop the Online Banking System, a robust set of technologies and tools will be utilized. Below is a comprehensive overview of the technology stack and tools that will be employed in the project.

3.1 Programming Language

- **Python**: Python is chosen for its simplicity, readability, and powerful libraries, making it well-suited for developing the banking system. Its extensive support for various functionalities allows for efficient coding and rapid development. Python's object-oriented programming (OOP) capabilities will be leveraged to ensure a modular and maintainable codebase. Additionally, Python's rich ecosystem of libraries and frameworks provides extensive resources for implementing the required features of the Online Banking System.

3.2 Integrated Development Environment (IDE)

- **Visual Studio Code**: Visual Studio Code (VS Code) is a versatile Integrated Development Environment (IDE) that is highly popular among developers for Python development. It offers a wide range of extensions and features, such as intelligent code completion and robust debugging tools, to enhance productivity and code quality. VS Code's seamless integration with version control systems, like Git, is essential for managing project source code and collaborating effectively with team members. Additionally, its extensive library of extensions allows developers to customize their environment with tools for testing, profiling, and debugging, facilitating a streamlined development process and ensuring the production of high-quality code.

3.3 Libraries and Frameworks

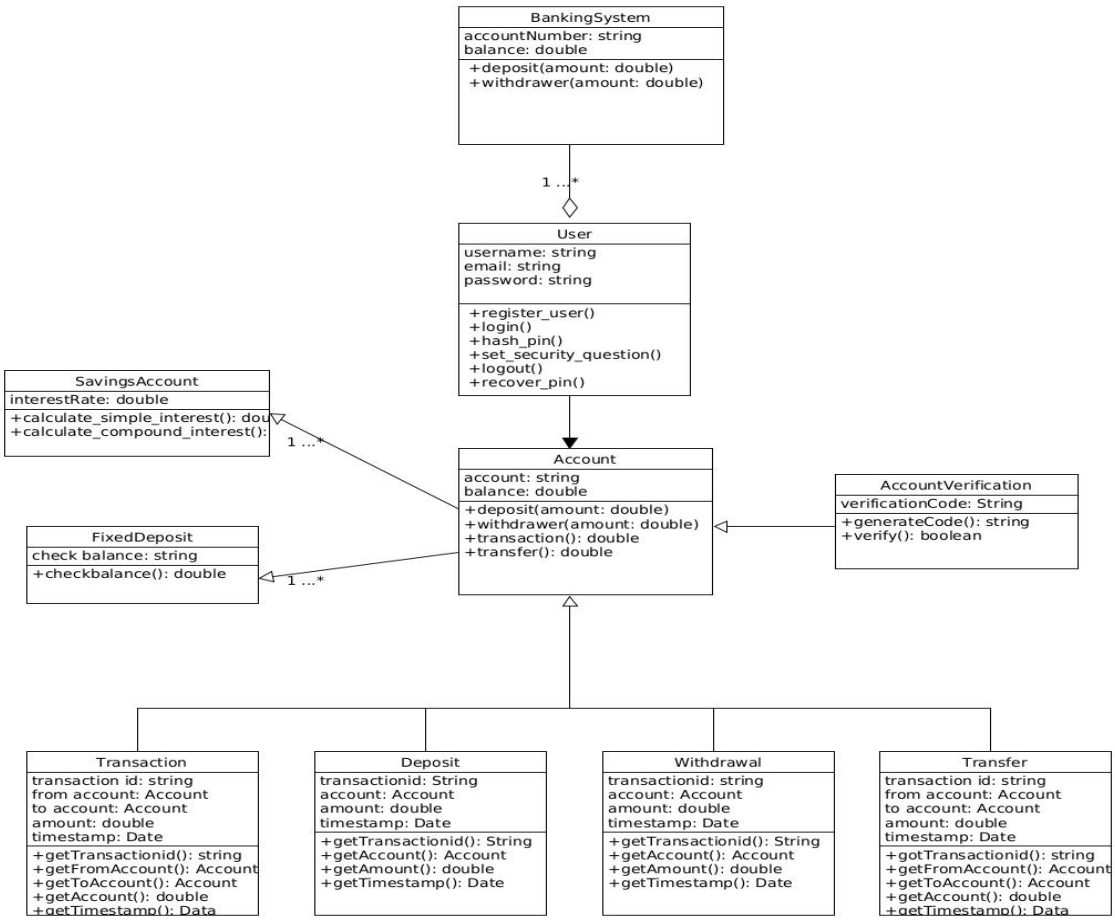
- **Math Library**: The Math library in Python will be utilized for performing necessary financial calculations such as interest rate calculations and compound interest computations. This library provides a wide range of mathematical functions that will be essential for implementing the banking system's core financial functionalities. Using the Math library ensures that the calculations are accurate and efficient, adhering to the required financial standards.

4. System Design

4.1 Class Diagram

Class diagrams are a crucial component of system design, particularly in Object-Oriented Programming (OOP). They visually represent the structure of a system by showing its classes, their attributes, methods, and the relationships between the classes. In the context of This application [online banking application], class diagrams serve several key purposes.

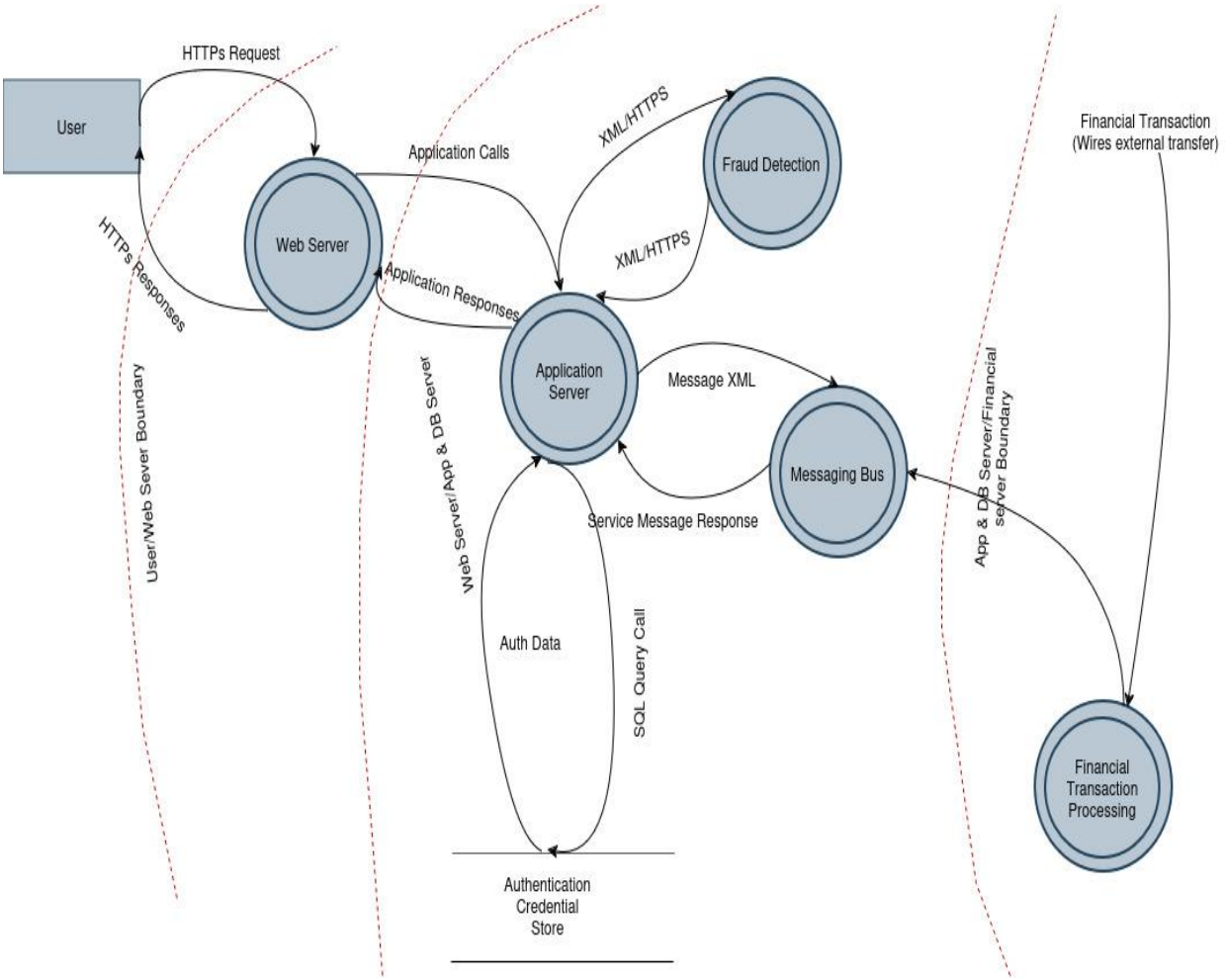
Class Diagram



4.2 Flow Diagram

The flow diagram for this application visually represents the technical process that users follow when interacting with the system. It typically includes technical steps like logging in, viewing account information, transferring funds etc and the internet protocols when this data travels.

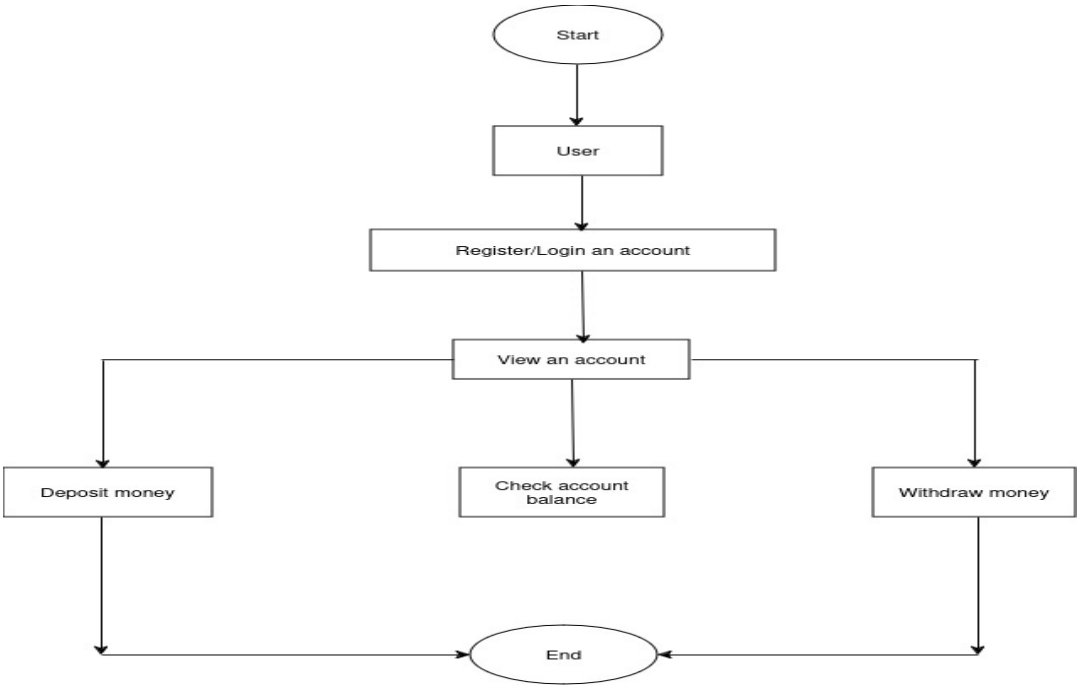
Flow Diagram



4.3 Algorithm Diagram

Algorithm diagram for this application represents the step-by-step logical flow of the application's key processes, such as user authentication, fund transfer, and other processes. It typically uses flowchart symbols to illustrate the decision points, processes, and data flow within the system as shown below.

Algorithm Diagram



5. Development

6. User Manual

6.1 Introduction

- **Purpose:**
  This user manual serves as a comprehensive guide for utilizing the Online Banking Application. It details the application's features, functionalities, and step-by-step instructions to ensure users can efficiently manage their financial transactions. The manual is designed to provide clear and concise guidance to help users navigate the system with ease, ensuring a seamless banking experience.
- **Audience:**
  This manual is intended for all end-users, including individuals, businesses, and organizations, who wish to leverage the Online Banking Application for secure and efficient management of their financial transactions. The users are expected to have basic computer literacy but no specific technical expertise is required to follow this guide.

6.2 Getting Started

Installation:

To begin using the Online Banking Application, users must first set up the necessary environment. This section outlines the steps required to properly install and configure the software on your system.

1. **Python Installation:**

- Step 1: Download the latest version of Python from the official website python.org. Ensure you download a version compatible with your operating system (Windows, macOS, or Linux).
- Step 2: Run the downloaded installer. During the installation process, make sure to check the box labeled "Add Python to PATH." This ensures that Python is accessible from the command line or terminal.
- Step 3: Follow the on-screen instructions to complete the installation. Once installed, you can verify the installation by opening a command prompt (Windows) or terminal (macOS/Linux) and typing python --version. This should display the installed Python version.

2. **Setting Up Visual Studio Code:**
   - Step 1: Download Visual Studio Code (VS Code) from the official website code.visualstudio.com. Ensure you select the appropriate version for your operating system.
   - Step 2: Run the installer and follow the installation prompts.
   - Step 3: After installation, launch Visual Studio Code. Navigate to the Extensions Marketplace by clicking on the square icon in the left sidebar and search for the "Python" extension. Install the official Python extension provided by Microsoft to enable Python support in VS Code.
   - Step 4: Configure the Python interpreter in VS Code. Open the Command Palette by pressing Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (macOS) and type Python: Select Interpreter. Choose the Python version you installed earlier.

Launching the Application: Once the environment is set up, you can launch the Online Banking Application by following these steps:

1. Step 1: Open Visual Studio Code.
2. Step 2: Use the file explorer in VS Code to navigate to the directory where the Online Banking Application code is stored.
3. Step 3: Open the main Python file (e.g., main.py) that contains the entry point for the application.
4. Step 4: Run the application by clicking on the "Run" button in the top-right corner of the editor, or by pressing F5. The application will start, and the command line interface (CLI) will prompt you to begin interacting with the banking system.
5. Step 5: Follow the on-screen instructions within the application to utilize its features, such as signing in, checking balances, and performing transactions.

6.3 User Registration

Sign Up

- Launch the application from your device.
- On the welcome screen, select the option to create a new account.
- Provide a unique username and set a secure 6-digit Personal Identification Number (PIN).
- Review your entered details carefully.
- Confirm your registration to complete the account creation process.

- Upon successful registration, a confirmation message will be displayed, and you will be redirected to the login page.

6.4 Logging In

Login Procedure

- Open the application and navigate to the login page.
- Enter your registered username and corresponding 6-digit PIN in the provided fields.
- Click the "Login" button to proceed.
- If the credentials are correct, you will be granted access to your account dashboard, where you can manage your banking activities.

6.5 Forgotten PIN

Resetting Your PIN:

- On the login page, select the "Forgot PIN" option.
- Enter your registered username to initiate the PIN recovery process.
- Follow the on-screen instructions to create a new 6-digit PIN.
- Confirm the new PIN by entering it again.
- Upon successful reset, you will be notified, and you can use the new PIN to log in.

6.6 Account Management

- Deposit Money:
    1. Choose the deposit option from the menu.
    2. Enter the amount to deposit.
    3. Confirm the transaction and check your updated balance.
- Withdraw Money:
    1. Select the withdrawal option.
    2. Input the amount you wish to withdraw.
    3. Confirm the withdrawal and view your remaining balance.
- Transfer Money**:**
    1. Choose the transfer option.
    2. Enter the destination account number and the amount to transfer.
    3. Confirm the transaction.

6.7 Checking Balance

- Viewing Your Balance:
    1. Select the balance inquiry option.
    2. Your current balance will be displayed.

6.8 Interest Rate and Compound Interest

- Deposit Interest Rate:

1. Choose the option to view your deposit interest rate.
2. The applicable rate based on your balance will be shown.

- **Ca**lculate Compound Interest:
    1. Select the compound interest calculation option.
    2. Choose to calculate based on your current balance or a specified amount.
    3. Input the number of years and confirm to see the future value.

6.9 Logging Out

- Exiting the Application:
    1. Select the option to log out.
    2. You will be securely logged out of your account.

6.10 Troubleshooting

- Common Issues:
    1. Incorrect PIN: Ensure the PIN entered is correct and retry.
    2. Invalid Account Number: Verify the destination account number and try again.
    3. Balance Issues: Check if your balance is sufficient for the transaction.

6.11 Code Documentation

```
import math

class User:
"""
A class to represent a bank user with basic functionalities such as deposit, withdraw, and transfer.

Attributes:
name (str): The name of the user.
pin (str): The user's 6-digit PIN code.
balance (float): The current balance of the user.
"""

 def __init__(self, name, pin, balance=0):
 """
Initializes a new User instance.
Args:
name (str): The name of the user.
pin (str): The 6-digit PIN of the user.
balance (float): The user's account balance. Default is 0.
"""
self.name = name
self.pin = pin
self.balance = balance



def deposit(self, amount):

"""
```

```
Deposits a specified amount into the user's account.
Args:
amount (float): The amount to be deposited.
Returns:
None
"""

        self.balance += amount

        print(f'Your current balance is {self.balance}')

def withdraw(self, amount):
"""
Withdraws a specified amount from the user's account.
Args:
amount (float): The amount to be withdrawn.
Returns:
None
"""
        if amount > self.balance:
                print('Your current balance is not available for transaction')
        else:
                self.balance -= amount
                print(f'{amount} has been withdrawn from your account and your current balance
is      {self.balance}')
def transfer(self, amount, dest_account):
"""
Transfers a specified amount to another account.
Args: amount (float): The amount to be transferred.
dest_account (str): The destination account number (must be 8 digits).
Returns:
None
 """
if len(dest_account) != 8:
    print('The transaction has been rejected since the destination account number is invalid')
elif amount > self.balance:
       print('Your current balance is not available for transaction')
else:
    self.balance -= amount
    print(f'The transaction of {amount} has been transferred to {dest_account} and your current balance is
{self.balance}')
def check_balance(self):
"""
Displays the user's current account balance.
Returns:
None
"""
print(f'Your current balance is {self.balance}')
def deposit_interest_rate(self):
 """
Determines the deposit interest rate based on the user's balance.
Returns:
None
"""
if self.balance > 50000:
    rate = 3
```

```
elif self.balance > 30000:
    rate = 2
else:
    rate = 1.5
print(f'Your current deposit interest rate is {rate} %')
def calculate_compound_interest(self, principal, rate, time):
"""
Calculates the compound interest for a given principal amount, rate, and time.
Args:
principal (float): The initial amount of money.
rate (float): The interest rate.
time (float): The number of years the money is invested or borrowed for.
Returns:
float: The amount after compound interest.
"""
a = principal * math.exp(rate * time)
return a
class BankingSystem:
"""
A class to represent the online banking system, which manages users, transactions, and account operations.
 """
def __init__ (self):
 """
Initializes a new BankingSystem instance with an empty user database.
Returns:
None
"""
self.users = {}
def signin(self):
"""
Signs in a new user by creating a username and 6-digit PIN, then adds them to the system.
 Returns:
None
 """
name = input('Please create your username: ')
pin = input('Please create your 6 digits pin: ')
if len(pin) != 6:
   print('The pin has to be 6 digits')
   self.signin()
else:
    self.users[name] = User(name, pin)
    print('Thanks for creating your account')
def forgotpin(self, user):
 """
Resets the user's PIN if they have forgotten it.
Args: user (User): The User object representing the user whose PIN needs to be reset.
Returns:
None
"""
new_pin = input('Please create your 6 digits pin: ')
if len(new_pin) != 6:
     print('The pin has to be 6 digits')
     self.forgotpin(user)
else:
      user.pin = new_pin
      print('The new pin has been reset, please login')
```

```
def login(self):
"""
Logs a user into the banking system if their username and PIN are correct.
Returns:
None
"""
name = input('Please enter your username: ')
pin = input('Please enter your pin: ')
user = self.users.get(name)
if user and user.pin == pin:
        print(f'Welcome to the online banking system, {name}')
            self.menu(user)
else:
print('Your username or password is wrong, did you create your account?')
self.create_or_reset_account()
def create_or_reset_account(self):
 """
Prompts the user to create an account or reset their PIN if they cannot log in.
Returns:
None
"""
choice = int(input('Enter 1 to create an account, 2 to reset your pin: '))
if choice == 1:
    self.signin()
elif choice == 2:
    name = input('Please enter your username to reset pin: ')
    user = self.users.get(name)
if user:
    self.forgotpin(user)
else:
    print('User not found. Please create an account first.')
    self.signin()
else: print('Option is unavailable')
def menu(self, user):
"""
Displays the banking system's menu to the logged-in user.
Args:
user (User): The User object representing the logged-in user.
Returns:
 None
"""
while True:
options = [
'1- Deposit',
'2- Withdraw',
 '3- Transfer',
 '4- Check Balance',
 '5- Deposit Interest Rate',
 '6- Calculate Compound Interest'
 ]
for option in options:
      print(option)
choice = int(input('Please enter the number provided above: '))
if choice == 1:
      amount = int(input('Enter the amount of your deposit: '))
```

```
        user.deposit(amount)
elif choice == 2:
        amount = int(input('Enter the amount of your withdrawal: '))
        user.withdraw(amount)
elif choice == 3:
        dest_account = input('Please enter the account number of your destination (8 digits): ')
        amount = int(input('Please enter the amount of money you want to transfer: '))
        user.transfer(amount, dest_account)
 elif choice == 4:
        user.check_balance()
elif choice == 5:
        user.deposit_interest_rate()
elif choice == 6:
        sub_options = [
          '1- Calculate your deposit compound interest based on your current balance',
          '2- Calculate your deposit compound interest based on your deposit input'
      ]
      for sub_option in sub_options:
            print(sub_option)
sub_choice = int(input('Please enter your choice from the options above: '))
if sub_choice == 1:
          time = float(input('Please enter how many years you want to calculate: '))
          rate = self.get_interest_rate(user.balance)
           print(f'Your current balance in {time} years will be{user.calculate_compound_interest(user.balance, rate,
time)}')
elif sub_choice == 2:
          time = float(input('Please enter how many years you want to calculate: '))
          principal = float(input('Please enter the amount of money in your current balance which you would like to
calculate: '))
        rate = self.get_interest_rate(principal)
        print(f'Your current balance in {time} years will be {user.calculate_compound_interest(principal, rate,
time)}')
else:
        print('Option is not available, back to the main menu')
def get_interest_rate(self, amount):
"""
Retrieves the interest rate based on the amount in the account.
Args: amount (float): The amount of money in the user's account.
Returns: float: The interest rate as a decimal.
 """
if amount > 50000:
      return 0.03
elif amount > 30000:
      return 0.02
else:
      return 0.015
if __name__ == "__main__":
      banking_system = BankingSystem()
       initial_option = int(input('Please choose 1 to sign in and 2 to log in: '))
      if initial_option == 1:
            banking_system.signin()
      elif initial_option == 2:
            banking_system.login()
      else:
            print('Option is unavailable')
      while True:
```

```
 another_transaction = int(input('Do you want to make another transaction? 1 for Yes, 2 for No: '))
if another_transaction == 1:
    banking_system.login()
elif another_transaction == 2:
    print('Thanks for using this online banking system')
    break
else:
    print('Option is unavailable')
```

7. Conclusion

7.1 Summary

The **Online Banking Application** project aimed to develop a secure, efficient, and user-friendly platform that enables users to manage their financial transactions online. Key features of the system include user registration, secure login, fund transfers, balance inquiries, interest rate calculations, and compound interest computation. By employing Object-Oriented Programming (OOP) principles and advanced data structures, this project provided an opportunity to integrate theoretical knowledge with practical application.

The system was developed using **Python** for its robust libraries and support for OOP concepts, with **Visual Studio Code** serving as the IDE. Features such as **user registration**, **deposit**, **withdrawal**, and **transaction handling** were successfully implemented. Security features like PIN recovery and validation mechanisms were included to enhance user experience and safeguard sensitive information. Through this project, the importance of secure and accessible digital banking solutions was highlighted, particularly in the current era of technological advancement.

7.2 Future Work

While the Online Banking Application has successfully met its objectives, several areas for improvement and expansion remain. Future work could include the following:

- **Mobile App Development**: Expanding the application to mobile platforms for enhanced accessibility and usability.
- **Two-factor Authentication**: Enhancing security by implementing two-factor authentication (2FA) for user logins.
- **Enhanced User Interface**: Improving the user interface with advanced design tools to offer a more visually appealing and intuitive experience.
- **Integration with External Banking APIs**: Connecting the application with real banking systems and APIs for real-time transactions and data synchronization.
- **Machine Learning for Fraud Detection**: Incorporating machine learning algorithms to detect and prevent potential fraudulent activities in real-time.

By pursuing these future developments, the system could offer even greater value to users and become more versatile in a real-world banking environment.

8.0 References

8.1 Bibliography

- Python.org. (n.d.). *Python Documentation*. Retrieved from https://docs.python.org
- Fowler, M. (2004). *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (3rd ed.). Addison-Wesley.
- Math Library Documentation. (n.d.). *Python Standard Library Documentation*. Retrieved from https://docs.python.org/3/library/math.html
- Downey, A. (2015). *Think Python: How to Think Like a Computer Scientist* (2nd ed.). O'Reilly Media.
- ISO/IEC 27001:2013. (n.d.). *Information security management standards*. International Organization for Standardization.

8.2 Related Documents

- Online Banking Security Guidelines, *Financial Conduct Authority*, Retrieved from https://www.fca.org.uk
- GDPR Compliance in Online Banking Applications, *European Union General Data Protection Regulation (GDPR)*, 2016.
- Open Banking Standards, *Open Banking Implementation Entity (OBIE)*, UK. Retrieved from https://www.openbanking.org.uk
- National Institute of Standards and Technology (NIST). (2019). *Cybersecurity Framework* (Version 1.1). Retrieved from https://www.nist.gov