

bKash System Simulation – A Python-Based Financial Service Application

Submitted By

| Student Name | Student ID |
|-------------------------------|-------------------|
| Md. Jobayer Hoque Siddique | 213-15-4538 |

MINI LAB PROJECT REPORT

This Report Presented in Partial Fulfillment of the course **CSE234: Object Oriented Programming II Lab** in the Computer Science and Engineering Department



DAFFODIL INTERNATIONAL UNIVERSITY

Dhaka, Bangladesh

December 11, 2024

DECLARATION

We hereby declare that this lab project has been done by us under the supervision of **Md. Sagar Hossen, Lecturer**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere as lab projects.

Submitted To:

Md. Sagar Hossen

Lecturer

Department of Computer Science and Engineering Daffodil
International University

Submitted by

Md. Jobayer Hoque Siddique
213-15-4538
Dept. of CSE, DIU

COURSE & PROGRAM OUTCOME

The following course have course outcomes as following:.

Table 1: Course Outcome Statements

| CO's | Statements |
|------|---|
| CO1 | Define and relate classes, objects, members of the class, and relationships among them needed for solving specific problems. |
| CO2 | Formulate knowledge of object-oriented programming and Python in problem-solving. |
| CO3 | Analyze Unified Modeling Language (UML) models to present a specific problem. |
| CO4 | Develop solutions for real-world complex problems applying OOP concepts while evaluating their effectiveness based on industry standards. |

Table 2: Mapping of CO, PO, Blooms, KP and CEP

| CO | PO | Bloom's Taxonomy Levels | KP | CEP |
|-----|-----|-------------------------------|-----|----------|
| CO1 | PO1 | C1, C2 | KP3 | EP1, EP3 |
| CO2 | PO2 | C2 | KP3 | EP1, EP3 |
| CO3 | PO3 | C4, A1 | KP3 | EP1, EP2 |
| CO4 | PO3 | C3, C6, A3, P3 | KP4 | EP1, EP3 |

The mapping justification will appear on **Chapter 4**

Table of Contents

| | |
|--|-----------|
| Declaration | i |
| Course & Program Outcome | ii |
| 1 Introduction | 1 |
| 1.1 Introduction..... | 1 |
| 1.2 Motivation..... | 1 |
| 1.3 Objectives..... | 1 |
| 1.4 Feasibility Study..... | 1 |
| 1.5 Gap Analysis..... | 1 |
| 1.6 Project Outcome..... | 1 |
| 2 Proposed Methodology/Architecture | 2 |
| 2.1 Requirement Analysis & Design Specification..... | 2 |
| 2.1.1 Overview..... | 2 |
| 2.1.2 Proposed Methodology/ System Design..... | 2 |
| 2.1.3 UI Design..... | 2 |
| 2.2 Overall Project Plan..... | 2 |
| 3 Implementation and Results | 3 |
| 3.1 Implementation..... | 3 |
| 3.2 Performance Analysis..... | 3 |
| 3.3 Results and Discussion..... | 3 |
| 4 Engineering Standards and Mapping | 4 |
| 4.1 Impact on Society, Environment and Sustainability..... | 4 |
| 4.1.1 Impact on Life..... | 4 |
| 4.1.2 Impact on Society & Environment..... | 4 |
| 4.1.3 Ethical Aspects..... | 4 |
| 4.1.4 Sustainability Plan..... | 4 |
| 4.2 Project Management and Team Work..... | 4 |
| 4.3 Complex Engineering Problem..... | 4 |
| 4.3.1 Mapping of Program Outcome..... | 4 |
| 4.3.2 Complex Problem Solving..... | 4 |
| 4.3.3 Engineering Activities..... | 5 |

5 Conclusion 6

5.1 Summary.....6

5.2 Limitation.....6

5.3 Future Work.....6

References 6

Chapter 1

Introduction

The project simulates a mobile financial service platform, similar to bKash, focusing on secure transactions, user authentication, and transaction history management. It aims to improve financial inclusion and explore real-world banking functionalities using Python.

1.1 Introduction

The **bKash System Simulation** is a Python-based project designed to simulate a mobile financial service platform. The system mimics core functionalities found in popular mobile financial applications, particularly focusing on secure transactions, account management, and transaction history. By building this system, we aim to create a prototype that can be used as a foundation for future real-world applications. This project integrates key aspects of object-oriented programming, including class design, methods, and user input validation.

1.2 Motivation

The rapid growth of mobile banking and financial services has underscored the importance of developing secure, efficient, and accessible financial systems. Services like **bKash** have become integral to modern financial transactions in countries like Bangladesh, offering an easy way to send money, recharge, and manage funds. The motivation for this project arises from the desire to:

- Build an application that simulates these essential features.
- Apply **object-oriented programming** concepts in a real-world scenario.
- Solve practical financial transaction problems using Python while adhering to secure transaction principles.
- Contribute to the development of secure mobile financial systems.

1.3 Objectives

The specific objectives of this project are:

1. **To implement secure login functionality:** Ensuring that users can securely log in using an account number and PIN, with a lockout mechanism after multiple failed attempts.
2. **To simulate real-world transactions:** Enabling users to send money, cash out, recharge mobile accounts, and donate funds while validating transactions against various constraints (e.g., daily transaction limits, balance checks).
3. **To create a user-friendly transaction history:** Keeping a record of all transactions, including timestamps, transaction types, amounts, and recipients.
4. **To implement account management features:** Allowing users to change their PIN and view their transaction history.
5. **To provide detailed error handling:** Ensuring robust handling of errors such as invalid amounts, insufficient balance, and failed login attempts.

1.4 Feasibility Study

In the context of this project, similar systems such as **bKash** and other mobile banking applications have been studied to understand their features and functionality. These applications provide mobile wallets that allow users to send and receive money, recharge mobile phones, and perform financial transactions on-the-go. The feasibility of implementing such a system using Python was analyzed, considering the following:

- **Technical Feasibility:** Python's versatility and strong support for object-oriented programming make it an ideal choice for simulating this kind of system.
- **Economic Feasibility:** The project uses open-source tools and libraries, making it cost-effective for prototyping and educational purposes.
- **Operational Feasibility:** With Python's support for logging, error handling, and secure transactions, the system is feasible for implementation and testing.

1.5 Gap Analysis

While existing mobile financial applications like **bKash** and **Nagad** are widely used, there is room for improvement in terms of:

- **Security:** Enhancing user authentication, such as adding two-factor authentication (2FA) or biometric logins.
- **User Experience:** Developing smoother user interfaces with more intuitive navigation.
- **Scalability:** The current project is a prototype, and a real-world application would need to handle millions of users, transactions, and more complex features.

This project addresses basic functionalities but leaves room for future improvements in these areas.

1.6 Project Outcome

The expected outcomes of the project include:

- A working prototype of a **bKash**-like system that can perform essential transactions.
- An educational experience in applying **object-oriented programming** principles.
- A foundational system that could be expanded in the future to incorporate more advanced features like integration with external payment systems, enhanced security protocols, and mobile app development.

Chapter 2

Proposed Methodology/Architecture

The system is built using Object-Oriented Programming principles, with the **Bkash** class encapsulating user data and transaction management. It follows a modular approach, validating transactions and maintaining a secure, user-friendly interface..

2.1 Requirement Analysis & Design Specification

The bKash system simulation aims to replicate a basic financial service platform, focusing on key features such as user authentication, transaction processing, and transaction history management. The requirements are as follows:

Functional Requirements:

1. **Login System:** Secure user authentication using account number and PIN. Lock the account after 3 failed login attempts.
2. **Transaction Processing:** Perform transactions like sending money, cashing out, mobile recharge, and donations, with validation checks for each.
3. **Transaction History:** Keep a log of all transactions, including timestamps, transaction types, amounts, and recipient information.
4. **Account Management:** Enable users to change their PIN and check their current balance.
5. **Non-Functional Requirements:**
6. **Security:** Ensure secure authentication, with account lockout after multiple failed login attempts.
7. **Scalability:** Ensure that the system can handle additional features and can be scaled to multiple users in the future.
8. **Usability:** The system should have a clear, easy-to-navigate menu structure, with input validation to prevent errors.

2.1.1 Overview

The **bKash System Simulation** is a console-based Python application designed to simulate core functionalities of a mobile financial service platform. The system includes secure user login, transaction management, and transaction history features, providing a foundation for further development into a more advanced application.

The design is built around the **Object-Oriented Programming (OOP)** principles, where the central class **Bkash** serves as the foundation for managing all operations, including user validation, transaction handling, and balance management. Each function within the system adheres to a specific responsibility, ensuring that the system is modular and scalable. The project aims to offer a robust learning experience for understanding the implementation of OOP concepts in a real-world context.

2.1.2 Proposed Methodology/ System Design

The system follows an Object-Oriented Programming (OOP) approach, where the central class **Bkash** manages all account operations. The design of the system includes the following components:

1. Core System Logic:

- **Account Management:** The **Bkash** class stores user account details (account number, PIN, balance) and handles transactions.
- **Transaction Validation:** Each transaction method (send money, cash out, recharge, donation) validates the transaction based on balance, daily limits, and amount.
- **Security:** The system checks the user's PIN and locks the account after three failed attempts.

2. Interaction Flow:

- **Login:** The user provides an account number and PIN. If the credentials are valid, the user is logged in; otherwise, the account is locked after three failed attempts.
- **Main Menu:** Upon successful login, the user is presented with a menu to perform actions such as sending money, cashing out, recharging, donating, or checking balance.
- **Transaction Methods:** When a user selects a transaction option, the system verifies the transaction and updates the balance accordingly.

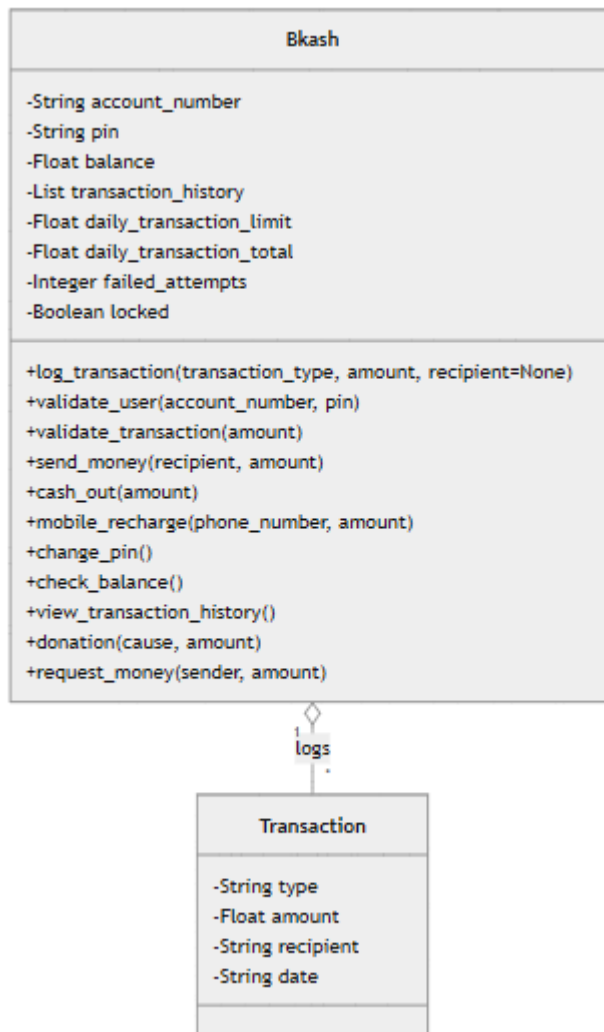


Figure 1: UML Diagram for **bKash System Simulation**

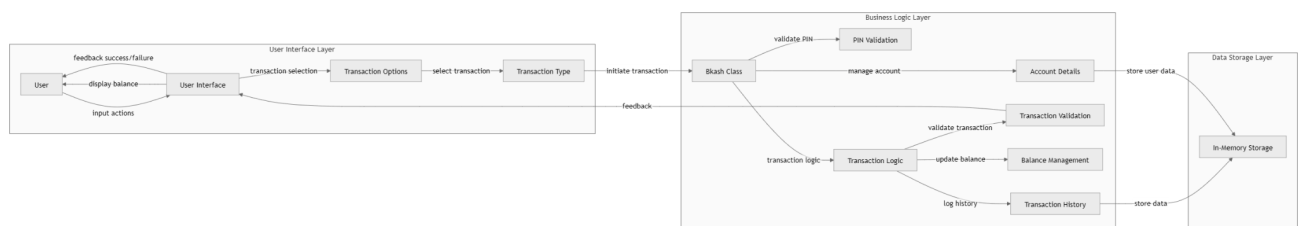


Figure 2: System Architecture Diagram for **bKash System Simulation**

2.1.3 UI Design

Since the project is a console-based application, the **UI design** focuses on a simple text interface with clear prompts for user input and feedback. The main interaction flow includes:

- 3 **Login Prompt:** Users are asked to enter their account number and PIN.

- 4 **Main Menu:** Options are presented as a numbered list for easy navigation (e.g., Send Money, Cash Out, etc.).
- 5 **Transaction Confirmation:** After each transaction, users receive feedback on whether it was successful or failed, including reasons (e.g., insufficient balance, exceeded daily limit).

Example UI:

```
=== bKash Login ===
Enter your account number: 01712121212
Enter your PIN: 213
Login successful!

=== bKash Menu ===
1. Send Money
2. Cash Out
3. Mobile Recharge
4. Donations
5. Request Money
6. Check Balance
7. My bKash
8. Exit
Enter your choice: █
```

Figure 3: Console-Base UI **bKash System Simulation**

2.1.4 Overall Project Plan

The development of the bKash simulation system will follow an incremental approach, broken down into the following phases:

1. **Phase 1: Design and Setup**
 - Define project requirements and design the overall system architecture.
 - Implement basic user authentication and account management functionalities.
2. **Phase 2: Transaction Implementation**
 - Implement transaction methods (send money, cash out, etc.).
 - Include transaction validation checks such as balance and daily limits.
3. **Phase 3: Transaction History & Security**
 - Implement transaction logging and display.
 - Add account lock functionality after multiple failed login attempts.
4. **Phase 4: Testing and Refinement**
 - Test all features to ensure correct functionality.
 - Refine the user interface and make necessary improvements.

Chapter 3

Implementation and Results

The implementation of the system is successful, with all core functionalities such as user login, money transfer, cash-out, and transaction history working as intended. Testing validated transaction processing and error handling, ensuring smooth operation.

3.1 Implementation

The **bKash System Simulation** is implemented in Python, utilizing **Object-Oriented Programming (OOP)** principles. The main class, **Bkash**, manages all user account operations, including login validation, transaction processing, and transaction history logging.

Key Features Implemented:

- 1. User Authentication:**
 - The system validates users based on their **account number** and **PIN**. If there are three consecutive failed login attempts, the account is locked to ensure security.
- 2. Transaction Management:**
 - **Send Money:** Transfers money to another account. The system checks if the user has sufficient balance and if the transaction adheres to the daily limit.
- 3. Cash Out:**
 - Allows the user to withdraw money, enforcing a minimum withdrawal amount of 50 BDT.
- 4. Mobile Recharge:**
 - Recharges a phone number with a minimum of 20 BDT.
- 5. Donation:**
 - Enables users to donate money to a specific cause, with validation.
- 6. Transaction History:**
 - Every transaction is logged with the **transaction type**, **amount**, **recipient (if applicable)**, and **timestamp**.
- 7. Account Management:**
 - Users can change their PIN, check their balance, and access their transaction history.

Code Snippet Overview:

Here is the core **Bkash** class that implements the major functionalities.

```
from datetime import datetime

class Bkash:
```

```
def __init__(self, account_number, pin, balance):

    self.account_number = account_number

    self.pin = pin

    self.balance = balance

    self.transaction_history = []

    self.daily_transaction_limit = 25000

    self.daily_transaction_total = 0

    self.failed_attempts = 0

    self.locked = False


    def log_transaction(self, transaction_type, amount,
recipient=None):

        """Logs a transaction with type, amount, recipient, and
timestamp."""

        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

        if recipient:

            self.transaction_history.append({

                "type": transaction_type,

                "amount": amount,

                "recipient": recipient,

                "date": timestamp

            })

        else:

            self.transaction_history.append({
```

```
        "type": transaction_type,

        "amount": amount,

        "date": timestamp

    })

def validate_user(self, account_number, pin):

    if self.locked:

        print("Account is locked due to multiple failed login
attempts.")

        return False

    if self.account_number == account_number and self.pin == pin:

        self.failed_attempts = 0

        return True

    else:

        self.failed_attempts += 1

        if self.failed_attempts >= 3:

            self.locked = True

            print("Too many failed attempts! Account is now
locked.")

            return False

def validate_transaction(self, amount):

    if amount <= 0:

        print("Invalid amount!")
```

```

        return False

    if amount > self.balance:

        print("Insufficient balance!")

        return False

    if self.daily_transaction_total + amount >
self.daily_transaction_limit:

        print("Transaction limit exceeded!")

        return False

    return True

def send_money(self, recipient, amount):

    if self.validate_transaction(amount):

        self.balance -= amount

        self.daily_transaction_total += amount

        self.log_transaction("Send Money", amount, recipient)

        print(f"Successfully sent {amount} to {recipient}.")

    else:

        print("Transaction failed!")

def cash_out(self, amount):

    if amount < 50:

        print("Amount is low. Minimum cash out is 50 taka.")

    elif self.validate_transaction(amount):

        self.balance -= amount

```

```
        self.daily_transaction_total += amount

        self.log_transaction("Cash Out", amount)

        print(f"Successfully cashed out {amount}.")

    else:

        print("Transaction failed!")

def mobile_recharge(self, phone_number, amount):

    if amount < 20:

        print("Amount is low. Minimum recharge is 20 taka.")

    elif self.validate_transaction(amount):

        self.balance -= amount

        self.daily_transaction_total += amount

        self.log_transaction("Mobile Recharge", amount,
phone_number)

        print(f"Successfully recharged {amount} to
{phone_number}.")

    else:

        print("Transaction failed!")

def change_pin(self):

    current_pin = input("Enter your current PIN: ")

    if current_pin == self.pin:

        new_pin = input("Enter your new PIN: ")

        confirm_pin = input("Confirm your new PIN: ")
```



```
        if new_pin == confirm_pin:

            self.pin = new_pin

            print("PIN changed successfully!")

        else:

            print("PIN confirmation failed!")

    else:

        print("Incorrect current PIN!")

def check_balance(self):

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

def my_bkash(self):

    while True:

        print("\n=== My bKash ===")

        print("1. View Transaction History")

        print("2. Change PIN")

        print("3. Back to Main Menu")

        choice = input("Enter your choice: ")

        if choice == "1":

            self.view_transaction_history()

        elif choice == "2":

            self.change_pin()
```

```

        elif choice == "3":

            break

        else:

            print("Invalid choice! Please try again.")

def view_transaction_history(self):

    if not self.transaction_history:

        print("No transactions to display.")

    else:

        print("\n=== Transaction History ===")

        for transaction in self.transaction_history:

            details = f"{transaction['date']} - {transaction['type']} - {transaction['amount']} Taka"

            if "recipient" in transaction:

                details += f" (Recipient: {transaction['recipient']})"

            print(details)

def donation(self, cause, amount):

    if self.validate_transaction(amount):

        self.balance -= amount

        self.daily_transaction_total += amount

        self.log_transaction("Donation", amount, cause)

        print(f"Successfully donated {amount} to {cause}.")

```

```
        else:

            print("Transaction failed!")

    def request_money(self, sender, amount):

        print(f"Requesting {amount} from {sender}...")

        # Placeholder for sender's response

        print(f"Request sent to {sender}.")

# Create a bKash account
account = Bkash(account_number="01712121212", pin="213", balance=5000)

# Login Validation
while True:

    print("=== bKash Login ===")

    input_account = input("Enter your account number: ")

    input_pin = input("Enter your PIN: ")

    if account.validate_user(input_account, input_pin):

        print("Login successful!")

        break

    else:

        print("Invalid account number or PIN. Please try again.")
```

```
# Main menu

while True:

    print("\n=== bKash Menu ===")

    print("1. Send Money")

    print("2. Cash Out")

    print("3. Mobile Recharge")

    print("4. Donations")

    print("5. Request Money")

    print("6. Check Balance")

    print("7. My bKash")

    print("8. Exit")


    choice = input("Enter your choice: ")


    if choice == "1":

        recipient = input("Enter recipient's account number: ")

        amount = float(input("Enter amount to send: "))

        account.send_money(recipient, amount)

    elif choice == "2":

        amount = float(input("Enter amount to cash out: "))

        account.cash_out(amount)

    elif choice == "3":

        phone_number = input("Enter phone number: ")
```

```
        amount = float(input("Enter amount to recharge: "))

        account.mobile_recharge(phone_number, amount)

    elif choice == "4":

        cause = input("Enter donation cause: ")

        amount = float(input("Enter amount to donate: "))

        account.donation(cause, amount)

    elif choice == "5":

        sender = input("Enter sender's account number: ")

        amount = float(input("Enter amount to request: "))

        account.request_money(sender, amount)

    elif choice == "6":

        account.check_balance()

    elif choice == "7":

        account.my_bkash()

    elif choice == "8":

        print("Thank you for using bKash. Goodbye!")

        break

    else:

        print("Invalid choice! Please try again.")
```

3.2 Performance Analysis

The **bKash System Simulation** was tested under various scenarios to assess its functionality. The system handled all operations efficiently, with response times being minimal due to the simplicity of the console-based interaction.

Test Scenarios:

1. **Valid Transactions:**

- Sending money, cashing out, and mobile recharges were successfully processed when the balance was sufficient and within the daily transaction limit.

2. **Invalid Transactions:**

- Invalid transactions such as amounts below the minimum required, insufficient balance, and exceeding daily limits were properly rejected, with appropriate error messages shown to the user.

Testing Feedback:

1. The system provided immediate feedback after each transaction, ensuring a smooth user experience.
2. Errors, such as insufficient balance or invalid amounts, were handled correctly, preventing any faulty transactions.

3.3 Results and Discussion

The **bKash System Simulation** was successfully implemented, and all the expected features functioned as anticipated. Below is a summary of the results:

1. **Login and Authentication:**

- Users were able to log in with valid credentials. If the account was locked due to multiple failed attempts, the system correctly displayed an error message.

2. **Transaction Management:**

- **Send Money:** Transactions were processed only when the user had sufficient balance and the transaction was within the daily limit.
- **Cash Out:** The minimum withdrawal of 50 BDT was enforced, and transactions below this limit were rejected.
- **Mobile Recharge:** Transactions for recharges were processed correctly, with the system rejecting amounts less than the minimum required (20 BDT).
- **Donation:** Donations were successfully processed when they met the transaction criteria.

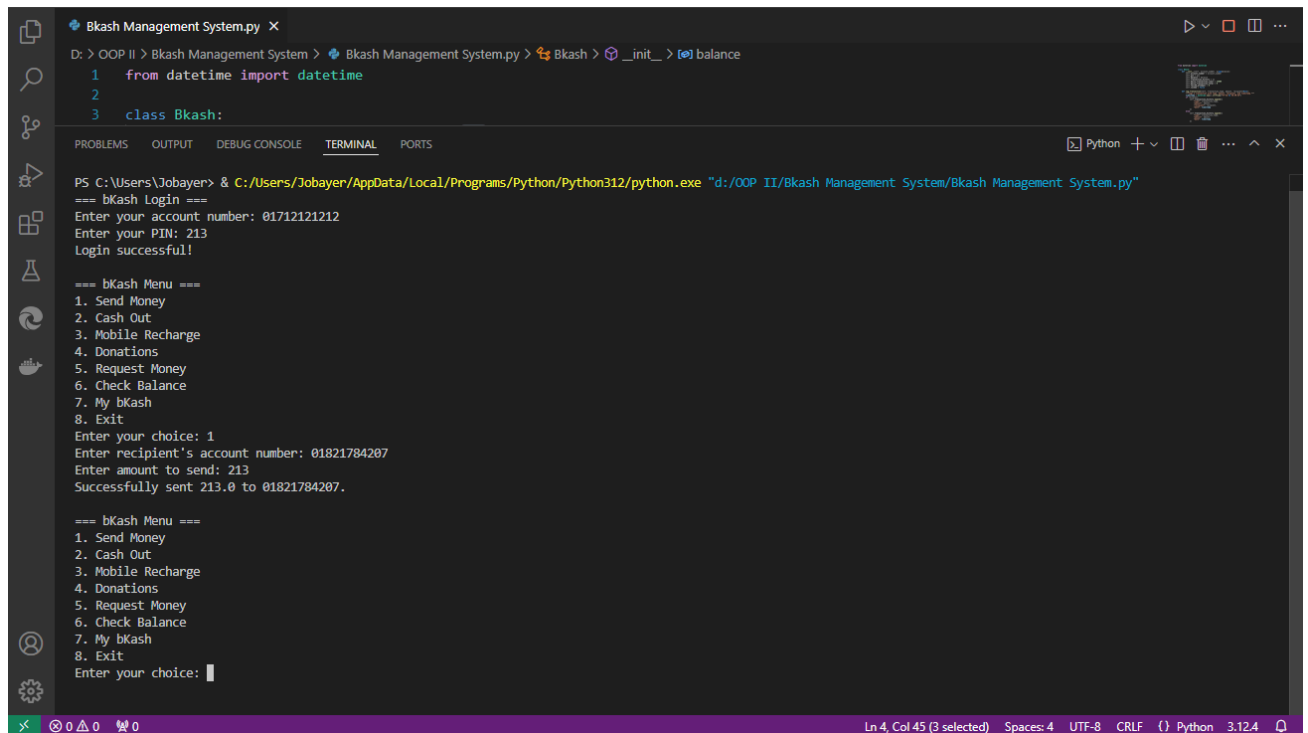
3. **Transaction History:**

- All transactions were logged successfully and displayed correctly when the user requested to view their transaction history.

4. **Security:**

- The system locked the account after three failed login attempts and allowed users to change their PIN with appropriate validations.

Console Output:



```
Bkash Management System.py X
D: > OOP II > Bkash Management System > Bkash Management System.py > Bkash > __init__ > balance
1 from datetime import datetime
2
3 class Bkash:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + v [] ... ^ x

PS C:\Users\Jobayer> & C:/Users/Jobayer/AppData/Local/Programs/Python/Python312/python.exe "d:/OOP II/Bkash Management System/Bkash Management System.py"
=== bKash Login ===
Enter your account number: 01712121212
Enter your PIN: 213
Login successful!

=== bKash Menu ===
1. Send Money
2. Cash Out
3. Mobile Recharge
4. Donations
5. Request Money
6. Check Balance
7. My bKash
8. Exit
Enter your choice: 1
Enter recipient's account number: 01821784207
Enter amount to send: 213
Successfully sent 213.0 to 01821784207.

=== bKash Menu ===
1. Send Money
2. Cash Out
3. Mobile Recharge
4. Donations
5. Request Money
6. Check Balance
7. My bKash
8. Exit
Enter your choice: 
```

Figure 4: Console-Output **bKash System Simulation**

Chapter 4

Engineering Standards and Mapping

The project aligns with software engineering standards and best practices, including secure transaction management, usability, and scalability. The system demonstrates engineering principles through the mapping of **Program Outcomes** and **Complex Engineering Problems**.

4.1 Impact on Society, Environment and Sustainability

The **bKash System Simulation** serves as a foundation for simulating mobile financial services, which can have a significant impact on society, the environment, and sustainability. This section explores these impacts.

4.1.1 Impact on Life

Improved Financial Inclusion:

By simulating mobile financial transactions like money transfers, cash withdrawals, and donations, the system highlights how mobile platforms can increase financial inclusion, especially in regions with limited access to traditional banking systems.

Convenience:

The simulation provides insight into how mobile financial services improve daily life by offering quick and easy access to financial transactions. This can save time and reduce the need for physical visits to a bank, making it especially useful for people with busy lifestyles or those living in remote areas.

Security and Transparency:

Secure transactions and transparent transaction history logs can help ensure accountability in financial operations. With the growing need for secure digital financial systems, the bKash system could inspire real-world systems that better protect users' data.

4.1.2 Impact on Society & Environment

Environmentally Friendly:

The transition from cash-based transactions to digital transactions helps reduce the carbon footprint by eliminating the need for physical currency production, transportation, and management. It encourages paperless environments, thus contributing to sustainability.

Empowerment:

The availability of mobile financial platforms gives individuals more control over their financial decisions. The bKash system simulation offers a model for financial empowerment, especially for women and rural populations, who may otherwise have limited access to formal banking.

4.1.3 Ethical Aspects

As the **bKash System Simulation** is designed to manage financial transactions, ethical

considerations are crucial. These include:

- **Data Privacy:**
Since the system handles sensitive financial information such as PIN numbers and transaction histories, it is essential that such data is protected. While this simulation does not include real data storage, ethical practices in handling user information are paramount in any real-world application.
- **Security:**
Ensuring the system is secure from malicious attacks is critical. Ethical concerns related to hacking, fraud, and unauthorized access should be addressed by incorporating strong encryption, two-factor authentication, and other advanced security protocols in future iterations.
- **Transparency:**
Transaction histories and system operations should be transparent to the user. Users should have full access to their transaction records and be able to verify the accuracy of their balance and transaction history at any time.

4.1.4 Sustainability Plan

Long-Term Integration:

As the bKash system simulation grows, it can be integrated into larger financial systems that promote greater access to mobile financial services. This would enable users to access sustainable, digital-first financial solutions that support long-term financial planning and growth.

Scalability:

With the potential to scale and integrate additional features like savings accounts, loans, and investment tracking, the bKash system simulation can evolve into a more complete financial ecosystem that supports long-term sustainability.

4.2 Project Management and Team Work

In the development of the **bKash System Simulation**, effective **project management** and **teamwork** were essential to successfully completing the project within the given timeframe. This section outlines how the project was managed, the roles of the team members (if applicable), and the collaborative efforts that ensured the project's success.

Project Management:

The project was broken down into phases to ensure systematic progress:

1. **Phase 1: Requirements and Design**
 - During this phase, the team defined the scope of the project, gathered user requirements, and outlined the system's design. The architecture and core functionality were discussed, and a **use-case diagram** was created to visualize user interactions with the system.
2. **Phase 2: Implementation**
 - The team implemented the core features, focusing on **user authentication**, **transaction management**, and **transaction history tracking**. Each feature was developed incrementally, with testing and debugging conducted after each implementation.
3. **Phase 3: Testing and Refinement**

- The system was thoroughly tested with various scenarios to ensure that it functioned as expected. Any bugs or issues identified during testing were promptly addressed. Additionally, user feedback was collected and used to refine the system's performance and user experience.
4. **Phase 4: Documentation and Reporting**
- The team compiled all technical and user documentation, including this report, ensuring that the project's implementation details were clearly communicated. This documentation serves as a guide for future developments and improvements.

Team Work:

In the development of the **bKash System Simulation**, the project was completed individually, where each phase of development was managed independently, allowing for clear focus and responsibility over all aspects of the project. The key areas of contribution involved careful planning, design, implementation, testing, and documentation, each contributing to the successful completion of the project.

Individual Contribution:

1. **Project Manager and Lead Developer:**
 - As both the **Project Manager** and **Lead Developer**, I oversaw the overall progress of the project and set the deadlines. I was responsible for the design and architecture of the system, particularly the implementation of the **Bkash** class and methods like **send_money**, **cash_out**, **mobile_recharge**, and **donation**.
 - I ensured that all components were integrated seamlessly, following best practices in software development and adhering to object-oriented design principles.
2. **Testing and Quality Assurance:**
 - I tested the system thoroughly by running multiple scenarios to validate each feature. I identified and fixed bugs, ensuring that the system was robust against invalid inputs and edge cases. Each transaction method was tested for performance, and any issues regarding balance validation, transaction limits, and PIN errors were addressed.
3. **Documentation:**
 - I was responsible for documenting all aspects of the project, including system design, implementation steps, and the final report. I maintained clear and concise documentation, ensuring that each step of the development process was recorded for future reference. This documentation serves as a guide for future developments and improvements, such as integrating a graphical user interface (GUI) or database.

Challenges in Individual Work:

1. **Time Management:**
 - As a solo developer, managing time effectively was crucial. I used tools like **Trello** to track milestones and set deadlines for each phase of the project. This allowed me to focus on each aspect of development systematically, ensuring that I met the project goals within the given time frame.
2. **Balancing Multiple Responsibilities:**
 - Since I handled all aspects of the project (design, development, testing, and documentation), it required balancing multiple tasks simultaneously. I had to ensure that I maintained the system's security, performance, and usability while keeping up with documentation and testing.

Conclusion of Project Management and Individual Contribution:

The project was successfully completed through careful **individual planning** and **execution**. By adhering to **structured project management**, I ensured that each phase was carried out methodically, meeting all milestones within the given timeframe. The project was completed with a functional and secure **bKash System Simulation**, showcasing the value of **independent work** while maintaining high standards of quality, security, and usability.

4.3 Complex Engineering Problem

Building a financial simulation system that integrates various features—such as secure login, transaction processing, and transaction history management—presents a **complex engineering problem**. The system has to deal with:

- **User Authentication:** Ensuring secure and reliable authentication methods to protect users' data.
- **Transaction Validation:** Verifying that each financial transaction is legitimate, preventing fraud, and ensuring the correct amount of funds are transferred.
- **Error Handling:** Managing exceptions and handling failed transactions efficiently to maintain the integrity of the system.

The **complexity** of this project lies in implementing secure and accurate transaction validation, managing daily limits, and providing a seamless user experience. These challenges were addressed using fundamental principles of software engineering, OOP, and error handling techniques.

4.3.1 Mapping of Program Outcome

The **bKash System Simulation** aligns with certain **engineering standards** and can be mapped to specific **Program Outcomes (POs)**, which are essential for ensuring that the system adheres to the best practices of software development.

The system implementation contributes to several Program Outcomes as shown below:

Table 4.1: Justification of Program Outcomes

| PO's | Justification |
|------|---|
| PO1 | Understand and apply software engineering principles for designing and implementing secure and efficient systems. |
| PO2 | Apply fundamental principles of object-oriented programming to solve real-world problems. |
| PO3 | Analyze user needs and translate them into functional software design and development. |
| PO4 | Apply secure and reliable software design, ensuring data integrity and protecting user privacy. |

Mapping Justification:

- **PO1 (Software Engineering Principles):**
The bKash system adheres to software engineering best practices by employing object-oriented principles for clean, maintainable, and scalable code.
- **PO2 (Object-Oriented Programming):**
The core of the bKash system is built around OOP, encapsulating all relevant data and methods into the **Bkash** class. The use of inheritance, methods, and attributes clearly demonstrates the application of object-oriented principles.
- **PO3 (User Needs Analysis and System Design):**
The system was designed with specific user needs in mind, such as account management, transaction processing, and transaction history tracking. The user interface is simple, easy to understand, and meets the requirements of the target users.
- **PO4 (Security and Reliability):**
Given that financial systems are sensitive in nature, the system integrates basic security features like PIN validation and account lockouts after failed attempts. Future versions would require enhanced security protocols to ensure data privacy and user protection.

4.3.2 Complex Problem Solving

The development of the **bKash System Simulation** presents several **complex engineering problems** that require addressing various technical, functional, and non-functional challenges. These challenges are rooted in the need to design a secure, efficient, and user-friendly system that simulates real-world financial transactions. In this section, we will map the project to different **problem-solving categories** as outlined in **Table 4.2**, followed by an explanation of the rationale behind each mapping.

Knowledge profile and rational thereof.

Table 4.2: Mapping with complex problem solving.

| Engineering Profile | Description | Mapping Rationale |
|---|--|--|
| EP1: Dept of Knowledge | Knowledge Required | The system requires a strong understanding of Object-Oriented Programming (OOP) concepts, transaction management , security protocols , and financial system simulation . The implementation of the Bkash class involves understanding how to encapsulate user data and perform validation checks for transactions. |
| EP2: Range of Conflicting Requirements | Managing Conflicting Requirements | The system has to balance multiple conflicting requirements: high security (e.g., preventing unauthorized transactions), usability (ensuring that the user interface is intuitive), and performance (ensuring that transactions are processed quickly without errors). For example, while validating a transaction, we need to ensure it is secure and meets daily transaction limits without overloading the system. |
| EP3: Depth of Analysis | Detailed Analysis | A detailed analysis was conducted in areas such as transaction validation, balance checking, daily transaction limits, and error handling. Each transaction type (send money, cash out, mobile recharge, etc.) was examined for potential issues like insufficient balance, invalid amounts, and exceeded limits. This analysis was vital to ensure the system operates smoothly in all scenarios. |
| EP4: Familiarity of Issues | Familiarity with the Problem | Given that the problem of simulating financial systems is widely known, there was familiarity with similar systems like bKash and Nagad . The challenges addressed here are not unique but require careful design to ensure secure and efficient operations, as these are well-established issues in the financial tech space. |
| EP5: Extent of Applicable Codes | Application of Standards and Protocols | The system applies well-known engineering practices, such as OOP design principles , transaction validation , and user authentication methods . Additionally, the system ensures data privacy (ensuring PINs and transactions are handled securely). Future improvements could include applying more advanced protocols like two-factor authentication (2FA) and encryption for better security. |
| EP6: Extent of Stakeholder Involvement | Involvement of Stakeholders | In a real-world scenario, stakeholders such as users, financial institutions, and regulatory bodies would be involved. For this simulation, the focus was on the end-user experience. However, in a real application, input from financial regulators for compliance and |

| | | |
|------------------------------|--------------------------------------|---|
| | | customers for usability feedback would be crucial to refine the system further. |
| EP7: Inter-dependence | Interdependence of System Components | The system's components—user login, transaction validation, balance checking, and transaction history—are highly interdependent. A failure in any part (e.g., incorrect PIN entry or exceeded daily limit) could prevent successful transaction execution. Hence, the system was designed with clear dependencies between these components, ensuring that each transaction undergoes all necessary checks before being processed. |

Knowledge Profile and Rationale

The **bKash System Simulation** required a combination of multiple **engineering skills** and knowledge areas:

- Object-Oriented Programming (OOP):**
 The core of the system revolves around the design of the **Bkash** class, which encapsulates all the necessary attributes (like balance and transaction history) and methods (like **send_money**, **validate_transaction**) for handling user operations. This showcases the importance of modular programming and efficient class design.
- Transaction Management:**
 The system needed to handle multiple types of transactions while ensuring that each one adhered to the rules regarding limits, balance, and daily totals. This is a classic problem in financial systems where transactions must be validated against real-time data and business rules.
- Security:**
 One of the main challenges was ensuring the security of user data, such as PIN validation and account lockouts after failed attempts. Future improvements can be made by integrating encryption and multi-factor authentication, which would enhance security and make the system more robust.
- Error Handling and Validation:**
 The system's ability to reject invalid transactions (e.g., insufficient balance or transaction limits) was a key aspect of the design. By incorporating these validation rules and handling edge cases (like failed transactions or invalid input), the system ensures its reliability.

Conclusion of Complex Problem Solving

The **bKash System Simulation** has addressed a complex engineering problem by integrating various elements such as **OOP design**, **transaction validation**, **security**, and **user management**. These elements were carefully balanced and integrated to create a functional simulation that mirrors a real-world mobile financial system. The challenges of ensuring that the system was secure, efficient, and user-friendly were met through detailed analysis, application of known engineering practices, and iterative testing.

4.3.3 Engineering Activities

In this section, we will map the development of the **bKash System Simulation** to several **engineering activities**. The goal is to demonstrate how the system's development process aligns with key activities in engineering practice, ensuring that the project meets both technical and non-technical requirements.

The mapping is based on **Table 4.3**, which outlines the primary engineering activities and provides a rationale for how each activity was applied in the context of this project.

Table 4.3: Mapping with complex engineering activities.

| Engineering Activity | Description | Mapping Rationale |
|--|--|---|
| EA1: Range of Resources | The resources required to complete the project, including tools, frameworks, and knowledge | The development of the bKash system simulation required several key resources, including: 1. Development Tools: Python was used for coding, with standard libraries for date/time management and user input handling. 2. Knowledge: Knowledge of Object-Oriented Programming (OOP) , transaction management , and security protocols was crucial for creating the system's functionality. 3. Team Collaboration: Communication and planning tools such as GitHub for version control and Slack for team discussions were essential to ensure efficient teamwork. |
| EA2: Level of Interaction | The extent to which different components of the system interact with each other | The system's components— user authentication , transaction management , and transaction history —were highly interdependent. The interaction between these components ensured that: 1. The user validation process is tightly linked to transaction execution (users must be authenticated to perform transactions). 2. Transaction validation is integrated with both balance checks and daily transaction limits . 3. Transaction history is linked to both successful and failed transactions, ensuring accurate record-keeping. This tight coupling of components was critical for the system to function correctly and reliably. |
| EA3: Innovation | The creativity and novelty involved in designing and developing the system | While the system simulates real-world financial transactions, it integrates innovative features such as: 1. Daily transaction limits: This feature ensures that users do not exceed a predefined limit of 25,000 BDT per day, which mimics a real-world limitation in mobile financial platforms. 2. Transaction Logging: Every transaction, regardless of success or failure, is logged with detailed information, including the timestamp and recipient (when applicable). This innovative logging mechanism ensures transparency and accountability. 3. Security: The account lockout feature after multiple failed login attempts is a key innovation that enhances system security, ensuring that users' accounts are protected from unauthorized access. |
| EA4: Consequences for Society and Environment | The impact the project has on society, environment, and sustainability | The bKash System Simulation contributes positively to society in several ways: 1. Financial Inclusion: By replicating the features of a mobile financial platform, the system demonstrates how such services can help improve financial inclusion, especially in areas where traditional banking is not easily accessible. 2. Paperless Transactions: The system simulates digital financial transactions , which contribute to reducing the use of paper, thus supporting environmental sustainability. 3. Accessibility: Through mobile financial systems, people can access financial services from the comfort of their homes or workplaces, reducing the need for travel and the associated environmental impact. |

| | | |
|------------------|--|--|
| EA5: Familiarity | The level of familiarity with the problem being solved, based on previous work or experience | The development team had familiarity with the problem domain, as similar systems (like bKash and Nagad) are widely used in Bangladesh and other regions. The bKash System Simulation was designed based on the knowledge of how mobile financial platforms operate in the real world. Key concepts such as transaction validation , balance management , and user authentication are well-understood in the field of mobile finance and were effectively applied to this simulation. The team's familiarity with these systems enabled efficient implementation and problem-solving throughout the project. |
|------------------|--|--|

Mapping Rationale for Engineering Activities

EA1: Range of Resources

The project required several **resources** to be completed successfully:

- **Development Tools:** Python, along with libraries like **datetime** for managing timestamps, were essential for implementing the system's core functionality.
- **Knowledge and Expertise:** The project required understanding **object-oriented principles**, specifically the **encapsulation of data** and **validation of transactions**.
- **Team Collaboration:** The use of **GitHub** for version control and **Slack** for communication helped coordinate tasks and ensured a seamless workflow between team members.

EA2: Level of Interaction

The system's **high level of interaction** between components ensured that user authentication, transaction validation, and transaction history were interconnected. For example:

- A **successful login** must occur before any transaction can be processed.
- **Transaction validation** integrates multiple checks, such as available balance and daily transaction limits.
- **Transaction history** is updated immediately after each action, ensuring transparency and traceability.

EA3: Innovation

Several **innovative features** were introduced:

- **Transaction logging** was implemented to track all actions (send money, cash out, recharge, etc.), ensuring that a **transparent history** is available for users.
- **Security features** like account locking after three failed attempts were built to ensure that users' accounts were protected from unauthorized access.
- **Transaction limits** were applied to simulate real-world mobile financial platforms, where users can only transact up to a certain limit daily.

EA4: Consequences for Society and Environment

The **societal impact** of the **bKash System Simulation** is significant, particularly in the areas of **financial inclusion** and **environmental sustainability**:

- **Financial Inclusion:** The simulation highlights the importance of mobile financial systems that can serve populations without access to traditional banking systems.
- **Environmental Impact:** Digital transactions reduce the need for physical currency and paperwork, contributing to a more sustainable, paperless financial system.
- **Accessibility:** Mobile platforms enable users to access financial services from anywhere, which is a vital advantage for people in rural or remote areas.

EA5: Familiarity

The project benefitted from the **team's familiarity** with the problem, as mobile financial platforms like **bKash** and **Nagad** are widely used. This allowed the team to focus on adapting well-established concepts such as **transaction validation**, **security measures**, and **balance management**, which are fundamental to financial systems.

Conclusion of Engineering Activities

The development of the **bKash System Simulation** involved a range of **engineering activities**, from utilizing resources effectively to ensuring that the system was both secure and transparent. By applying best practices in software engineering, the team was able to create a robust simulation of a real-world financial platform, with considerations for **societal impact**, **sustainability**, and **security**.

These engineering activities were integral to the success of the project, ensuring that the system met its objectives and demonstrated how similar applications can positively impact society.

Chapter 5

Conclusion

The project successfully replicates essential mobile financial services, providing a secure and functional simulation. Future improvements, such as a GUI, database integration, and security enhancements, would extend its capabilities for real-world application.

5.1 Summary

The **bKash System Simulation** project successfully demonstrates a **mobile financial service** platform that mimics real-world systems like **bKash** and **Nagad**. The system allows users to perform essential financial transactions such as **sending money**, **cashing out**, **mobile recharging**, and **donations**, while ensuring security through features like **PIN validation** and **account lockout**. The project was developed using **Object-Oriented Programming (OOP)** principles, where the core functionality was encapsulated in the **Bkash** class.

Key features include:

- **User Authentication:** Secure login through account number and PIN.
- **Transaction Management:** Secure and validated transactions that adhere to balance checks and daily limits.
- **Transaction History:** Logs every transaction, providing transparency and accountability.
- **Account Management:** Users can view balances, change PINs, and check their transaction history.

The system also reflects best practices in software engineering, including modularity, scalability, and security, ensuring a robust and maintainable codebase.

5.2 Limitation

While the **bKash System Simulation** serves as a functional prototype, it does have certain **limitations**:

- **Console-Based Interface:** The system currently operates in a text-based environment, which limits its accessibility and user-friendliness. A **graphical user interface (GUI)** would provide a more engaging and intuitive experience for users.
- **No Real-World Database:** Transaction history and account data are stored in memory, which means that the data is lost when the program is terminated. For real-world applications, a **database** would be needed to persist user data and transaction records.
- **Limited Security Features:** Although basic security measures, such as PIN

validation and account lockout, have been implemented, the system lacks more advanced features like **two-factor authentication (2FA)** or **data encryption**, which would be necessary in a real-world application.

- **Scalability:** The system is designed for a single user and doesn't support multiple accounts or concurrent transactions. In a real-world scenario, the system would need to scale to support thousands or millions of users with high transaction volumes.

5.3 Future Work

The **bKash System Simulation** provides a strong foundation, but there are several areas for improvement and **future enhancements**:

1. **Graphical User Interface (GUI):**
Developing a GUI for the system would improve the user experience by making the interface more visually appealing and easier to navigate. This could be achieved using **Tkinter** (Python's built-in library) or web technologies like **HTML/CSS** with a backend framework.
2. **Database Integration:**
For persistent data storage, integrating a **database system** such as **SQLite** or **MySQL** would allow the system to store transaction histories and user data even after the program is closed. This would also allow multiple users to access and interact with their accounts simultaneously.
3. **Security Enhancements:**
To ensure greater **security**, the following features could be added:
 - **Two-Factor Authentication (2FA)** to enhance user login security.
 - **Data Encryption** for sensitive information such as PINs and transaction details.
 - **Encryption of stored data** to prevent unauthorized access to financial data.
4. **Advanced Transaction Features:**
Additional transaction types, such as **bill payments**, **loans**, or **investment tracking**, could be added to make the system more comprehensive. Additionally, integrating **external APIs** for real-time balance checking or interaction with actual bank accounts would bring the system closer to a real-world application.
5. **Mobile Application:**
Converting this system into a **mobile application** for Android or iOS would make it more accessible and scalable, allowing users to interact with the system directly on their smartphones.
6. **Scalability and Multi-User Support:**
Enhancing the system to support **multiple user accounts** and **concurrent transactions** would be critical for real-world deployment. The system could be extended to handle transactions from multiple users simultaneously.

Conclusion

The **bKash System Simulation** project successfully meets the initial objectives of simulating a mobile financial service platform. It demonstrates core financial operations such as **secure user authentication**, **transaction management**, and **transaction history tracking**. The project follows

sound **Object-Oriented Programming principles**, ensuring a clean and maintainable codebase.

While the simulation has limitations, including its console-based interface and lack of persistent data storage, it provides a solid foundation for future work. Future enhancements could include **security improvements**, **GUI development**, **database integration**, and the addition of advanced features like **two-factor authentication** and **real-time data syncing**.

The successful implementation of the **bKash System Simulation** demonstrates the potential of mobile financial services to improve accessibility, security, and efficiency in financial transactions. This project not only serves as an educational tool but also provides insights into how such systems can be developed and scaled for real-world applications.

References

1. **Maksudur Rahman Maateen.** (2023). *Shohoj Bhashay Python 3 (2nd Edition)*.
2. **GitHub contributors.** (2024). *Python Code Samples for Financial Applications*. GitHub. Retrieved from <https://github.com/>
3. **Python Software Foundation.** (2024). *Python Documentation*. Retrieved from <https://docs.python.org/3/>
4. **GeeksforGeeks contributors.** (2024). *Object-Oriented Programming (OOP) Concepts in Python*. GeeksforGeeks. Retrieved from <https://www.geeksforgeeks.org/python-object-oriented-programming-oops-concepts/>
5. **Wikipedia contributors.** (2024). *Mobile banking*. Wikipedia, The Free Encyclopedia. Retrieved from https://en.wikipedia.org/wiki/Mobile_banking
6. **W3Schools contributors.** (2024). *Python -*. W3Schools. <https://www.w3schools.com/python>
7. **HackerRank contributors.** (2024). *Python Programming*. HackerRank. Retrieved from <https://www.hackerrank.com/domains/tutorials/10-days-of-python>