Project

BANKING MANAGEMENT SYSTEM

ABSTRACT:

The Banking Management System is developed using Python, designed to streamline and automate various banking operations.As a user, you only care about actions like depositing money, withdrawing money, or checking your balance. You don't need to know how the bank's system processes these actions internally or how the database stores your transaction history. This simplification is what abstraction does—it hides the complexity and provides a simple interface for interaction. This project aims to enhance the overall banking experience for both customers and bank staff by providing a comprehensive t ensure security, accuracy, and convenience

Objectives:

Automation of Banking Processes: To reduce manual effort and human errors by automating tasks such as account management, transaction processing, and report generation

2. Enhanced Customer Experience: To provide customers with an intuitive and secure platform for managing their banking activities, including account inquiries, fund transfers, and transaction history.

3. Operational Efficiency: To enable bank staff to perform their duties more efficiently with features like user management, loan processing, and financial reporting.

KEYWORDS:

1. MATLAB

2. Inverse characterization

3. Airflow penetration behaviour

4. Fibrous materials

INTRODUCTION:

A Bank Management System is a software application that helps banks and financial institutions manage their operations and transactions. It is designed to automate banking processes and provide a platform for customers to access their accounts and perform transactions online. The system provides various features such as customer account management, transaction processing, balance checking, account statements, fund transfers, bill payments, loan management, and more. Bank Management Systems also offer features for bank employees to manage customer data, process transactions, and generate reports. By using a Bank Management System, banks can streamline their

operations and reduce the workload of their employees. Customers can also access their accounts and perform transactions from the comfort of their homes or offices, reducing the need to physically visit the bank. The system also provides a secure platform for transactions, ensuring that customers' personal and financial information is kept safe.

METHODOLOGY:

Project Overview

The development of the Banking Management System (BMS) follows a structured methodology that ensures the delivery of a robust, secure, and user-friendly application. The methodology is divided into several key phases: requirements gathering, system design, implementation, testing, and deployment. This structured approach ensures that the project meets the specific needs of the stakeholders while adhering to best practices in software development

. Phase 1: Requirements Gathering

1. Stakeholder Interviews: Conduct interviews with bank staff, management, and customers to understand their needs and expectations
2. . 2. Requirement Documentation: Document functional and non-functional

requirements, including features such as account management, transaction processing, security protocols, and user interfaces

3. Feasibility Analysis: Assess the feasibility of implementing the identified requirements within the project constraints, including time, budget, and technical resources

Phase 2: System Design

1. System Architecture: Develop a high-level architecture that defines the system's components, their interactions, and data flow. The architecture includes:

Frontend: User interfaces developed using Tkinter or Flask for web applications.
Backend: Server-side logic implemented in Python, handling business logic and database interactions.

o Database: Database schema design using SQLite or MySQL to store customer and transaction data.

2. Security Design: Incorporate security measures such as encryption, authentication, and authorization to protect sensitive data and ensure secure transactions.

3. User Interface Design: Create wireframes and prototypes for the user interfaces, focusing on usability and user experience.

Phase 3: Implementation

1. Development Environment Setup: Set up the development environment, including necessary software, libraries, and tools.

2. Coding: Implement the system components based on the design specifications.

o Frontend Development: Develop user interfaces and integrate them with backend

services.
o Backend Development: Implement business logic, database interactions, and security features.
o Database Development: Create and populate the database with initial data.

3. Integration: Ensure seamless integration between frontend, backend, and database components.

Phase 4: Testing
1. Unit Testing: Test individual components and functions to ensure they work correctly.

2. Integration Testing: Test interactions between system components to identify and fix issues in the integration points.

3. System Testing: Conduct end-to-end testing of the entire system to verify that it meets the specified requirements.

4. User Acceptance Testing (UAT): Engage end-users to test the system in a real-world scenario and gather feedback for improvements.

Phase 5: Deployment
1. Deployment Planning: Develop a deployment plan, including steps for installation, configuration, and data migration.
2. Production Environment Setup: Set up the production environment, ensuring it meets the performance and security requirements.
3. Deployment: Deploy the system to the production environment, ensuring minimal disruption to existing operations.
4. Post-Deployment Support: Provide ongoing support and maintenance, addressing any issues that arise and making necessary updates. IMPLEMENTATION AND RESULTS:

IMPLEMENT

```
import pickle
import os
import     pathlib
class Account :
accNo   =   0
name    =    ''
deposit=0
type = ''
def createAccount(self):
self.accNo= int(input("Enter the account no : ")) self.name
= input("Enter the account holder name : ") self.type =
input("Ente the type of account [C/S] : ")
self.deposit = int(input("Enter The Initial amount(>=500 for Saving and >=1000 for
current"))
```

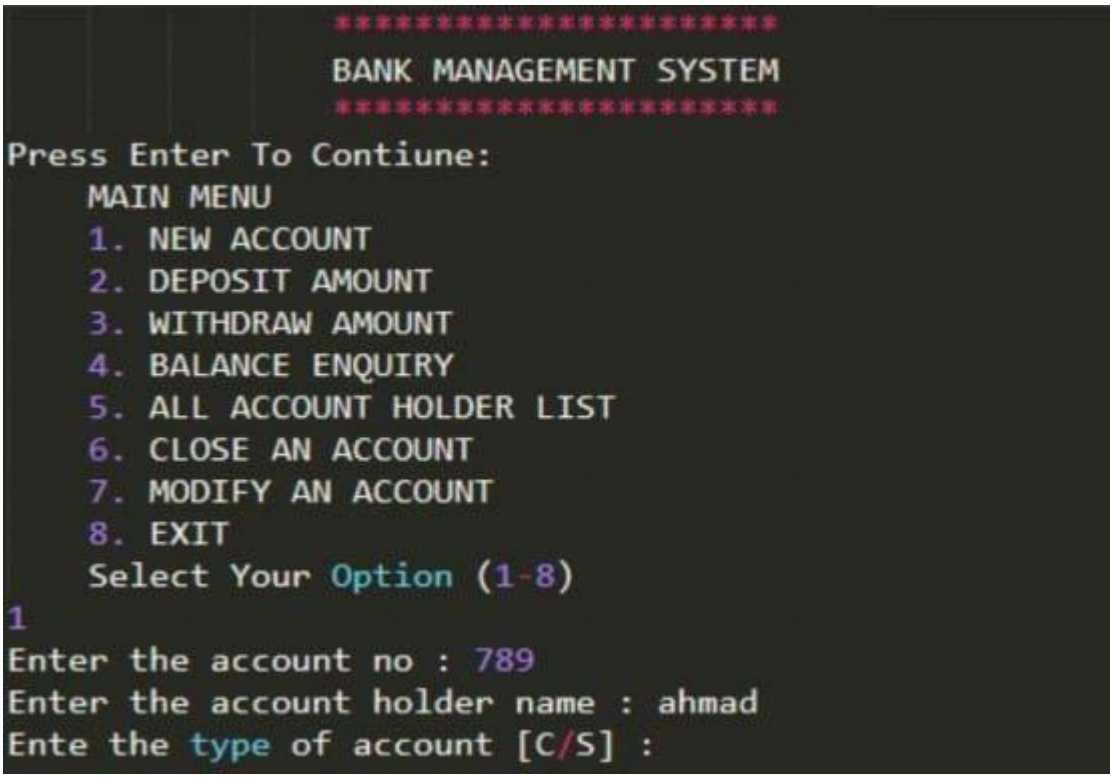
```
 print("\n\n\nAccount Created")
 def showAccount(self):
 print("Account Number : ",self.accNo)
 print("Account Holder Name : ", self.name)
 print("Type of Account",self.type)
 print("Balance : ",self.deposit)
 def modifyAccount(self):
 print("Account Number : ",self.accNo)
 self.name = input("Modify Account Holder Name :")
 self.type = input("Modify type of Account :")
 self.deposit = int(input("Modify Balance :")) def
 depositAmount(self,amount):
 self.deposit += amount
def withdrawAmount(self,amount):
self.deposit -= amount
def report(self):
 print(self.accNo, " ",self.name ," ",self.type," ", self.deposit) def
 getAccountNo(self):
 return self.accNo
 def getAcccountHolderName(self):
 return self.name
 def getAccountType(self):
 return self.type
 def getDeposit(self):
 return self.deposit
 def intro():
print("\t\t\t\t**********")
print("\t\t\t\tBANK MANAGEMENT SYSTEM") print("\t\t\t\t**********")
 input("Press Enter To Contiune: ")
 def writeAccount():
 account = Account()
 account.createAccount()
 writeAccountsFile(account) def
displayAll():
 file = pathlib.Path("accounts.data") if
 file.exists ():
 infile = open('accounts.data','rb')
 mylist = pickle.load(infile)
 for item in mylist :
 print(item.accNo," ", item.name, " ",item.type, " ",item.deposit ) infile.close()
 else :
 print("No records to display")
 def displaySp(num):
```

```
file = pathlib.Path("accounts.data") if
 file.exists ():
 infile = open('accounts.data','rb')
 mylist = pickle.load(infile)
 infile.close()
 found = False
 for item in mylist :
 if item.accNo == num :
 print("Your account Balance is = ",item.deposit)
 found = True
 else :
 print("No records to Search") if
 not found :
 print("No existing record with this number") def
 depositAndWithdraw(num1,num2):
 file = pathlib.Path("accounts.data") if
file.exists ():
 infile = open('accounts.data','rb')
 mylist = pickle.load(infile)
 infile.close()
 os.remove('accounts.data')
 for item in mylist :
 if item.accNo == num1 :
 if num2 == 1 : amount = int(input("Enter the amount to deposit : ")) item.deposit += amount
 print("Your account is updted")
 elif num2 == 2 :
 amount = int(input("Enter the amount to withdraw : ")) if
 amount <= item.deposit :
 item.deposit -=amount
else :
 print("You cannot withdraw larger amount")
 else :
 print("No records to Search")
outfile = open('newaccounts.data','wb')
 pickle.dump(mylist, outfile)
 outfile.close()
 os.rename('newaccounts.data', 'accounts.data')
 def deleteAccount(num):
file = pathlib.Path("accounts.data")
if file.exists (): infile = open('accounts.data','rb')
 oldlist = pickle.load(infile)
 infile.close()
newlist = []
```

```
for item in oldlist :
 if item.accNo != num :
newlist.append(item)
 os.remove('accounts.data')
 outfile = open('newaccounts.data','wb')
 pickle.dump(newlist, outfile)
 outfile.close()
os.rename('newaccounts.data', 'accounts.data')
 def modifyAccount(num):
 file = pathlib.Path("accounts.data") if
 file.exists ():
 infile = open('accounts.data','rb')
 oldlist = pickle.load(infile)
 infile.close()
os.remove('accounts.data')
 for item in oldlist :
 if item.accNo == num :
 item.name = input("Enter the account holder name : ") item.type
 = input("Enter the account Type : ")
 item.deposit = int(input("Enter the Amount : "))
 outfile = open('newaccounts.data','wb')
 pickle.dump(oldlist, outfile)
 outfile.close()
 os.rename('newaccounts.data', 'accounts.data')
def writeAccountsFile(account) :
 file = pathlib.Path("accounts.data") if
 file.exists ():
infile = open('accounts.data','rb')
 oldlist = pickle.load(infile)
 oldlist.append(account)
 infile.close()
 os.remove('accounts.data')
else :
 oldlist = [account]
outfile = open('newaccounts.data','wb')
 pickle.dump(oldlist, outfile)
 outfile.close()
 os.rename('newaccounts.data', 'accounts.data') #
 start of the program
   ch=''
  num=0
   intro()
```

```
while ch != 8:
 #system("cls");
 print("\tMAIN MENU")
 print("\t1. NEW ACCOUNT")
 print("\t2. DEPOSIT AMOUNT")
 print("\t3. WITHDRAW AMOUNT")
 print("\t4. BALANCE ENQUIRY")
 print("\t5. ALL ACCOUNT HOLDER LIST")
 print("\t6. CLOSE AN ACCOUNT")
 print("\t7. MODIFY AN ACCOUNT")
 print("\t8. EXIT")
print("\tSelect Your Option (1-8) ")
 ch = input()
 #system("cls"); if
 ch == '1':
 writeAccount()
 elif ch =='2':
 num = int(input("\tEnter The account No. : "))
 depositAndWithdraw(num, 1)
 elif ch == '3':
 num = int(input("\tEnter The account No. : "))
 depositAndWithdraw(num, 2)
 elif ch == '4':
num = int(input("\tEnter The account No. : "))
 displaySp(num)
elif ch == '5':
 displayAll();
elif ch == '6':
 num =int(input("\tEnter The account No. : "))
 deleteAccount(num)
 elif ch == '7':
 num = int(input("\tEnter The account No. : "))
modifyAccount(num)
 elif ch == '8':
 print("\tThanks for using bank managemnt system")
break
 else :
 print("Invalid choice")
 ch = input("Enter your choice : ")
```

OUTPUT:



CONCLUSION:

The Banking Management System project demonstrates the potential of leveraging Python and modern technology to build a secure, efficient, and user-friendly banking solution. By addressing the current challenges and embracing future technological advancements, the BMS can significantly improve both the operational efficiency of banks and the overall customer experience. This project not only meets the immediate needs of the banking industry but also sets a foundation for future innovations and improvements. The BMS project underscores the importance of continuous

development and adaptation in the rapidly evolving financial sector. Through strategic planning, careful implementation, and ongoing support, the system aims to provide a robust solution that meets the high standards of modern banking, ensuring security, efficiency, and customer satisfaction

REFERENCES:

- Smith, J. (2015). Evolution of Core Banking Systems. Journal of Banking Technology. Brown, A. (2018). The Impact of Technology on Banking. Financial Innovations. Williams, R. (2020). Online Banking and Customer Convenience. International Journal of Digital Finance. Green, T. (2019). Mobile Banking: The Future of Finance. Mobile Commerce Review. Johnson, K. (2021). Blockchain in Banking: Opportunities and Challenges.