PM SHRI

Pradhan Mantri ScHools for Rising India

Creating future ready and globally competent individuals equipped with key 21st Century skills

तत् त्वं पूषन् अपावृणु

केन्द्रीय विद्यालय संगठन

PM SHRI
KENDRIYA VIDYALAYA PANAGARH
2024-25

SUBJECT – INFORMATICS PRACTICES

PROJECT TOPIC – BANK MANAGEMENT SYSTEM

NAME – ADITYA KUMAR GOND

CLASS – 12

ROLL NO –

STREAM – COMMERCE

Certificate

This is to certify that Aditya Kumar Gond a student of class 12 has successfully completed his Informatics Practices project titled "Bank Management System" under the guidance of Mrs. Tanusree Sadhukhan. This project, which makes use of MySQL and Python, shows excellent dedication, hard work, and technical skill. This certificate is given in appreciation of the outstanding work and achievements that were presented in the project.

_____
Signature of external examiner

_____
Signature of internal examiner

_____
Signature of Principal

Acknowledgement

I would like to express my sincere thanks to everyone who helped me in this project. My project guide, Mrs. Tanusree Sadhukhan, was really helpful in shaping the project and improving my understanding with her knowledge and support.

I would like to thank my peers and colleagues for their valuable help and support in improving and overcome difficulties with the idea. I thank the PM Shri Kendriya Vidyalaya Panagarh faculty and staff for their invaluable resources.

Finally, I would like to thank my friends and family for their help and support in this project. The project's success was greatly dependent on their efforts.

Bank Management System

Index

Abstract

In this Python script, a comprehensive Bank Management System is implemented using MySQL as the backend database. A number of features are available, including account creation, fund transfers, balance checks, account listing, and account closure.

Key Components:

1. **Database Connectivity:** Easily communicates with a MySQL database that is hosted locally.
2. **Account Operations:** Opening Accounts: Provide the name, type of account (savings or current), and amount of the initial deposit.
3. Safe transactions with balance confirmation are made when depositing and withdrawing money.
4. **Balance Inquiry:** Use your account number to check your balance.
5. **Managing Accounts:** Listing Accounts View every account's details.
6. **Closing Accounts:** Use the account number to close accounts.
7. **User Interface:** 1–7 options on a basic CLI.

In summary:

This system offers the fundamental banking features, showcasing real-world database management and user interface applications, providing a strong foundation for additional enhancements. Ready to Get Started?

System requirements

Hardware Requirements:

1. Processor: intel core i3 or Intel core i5
2. RAM: 4 GB or more
3. Storage: At least 4GB of free disk space
4. Display: 1024 x 768 resolution or higher

Software Requirements:

1. Operating System:
    - Windows 10 or 11
    - macOS 10.12 or higher
    - Linux (any modern distribution)
2. Python: Version 3.6 or higher
3. MySQL: Version 5.7 or higher

Python Libraries:

1. mysql-connector-python
2. random

Feasibility Study

The feasibility study assesses the Bank Management System's viability, considering technical and economic aspects. It evaluates development potential, operational costs, and benefits, informing stakeholders to aid decision-making and resource allocation.

Technical Feasibility:

The Bank Management System shows strong technical feasibility, using widely-supported Python and MySQL. Modest hardware needs allow deployment on standard infrastructure. Python's simplicity and MySQL's reliability, along with mysql-connector-python, enhance its technical feasibility.

Economic Feasibility:

The Bank Management System is economically feasible, with minimized development costs using open-source technologies. Low operational costs and minimal training expenses make it an attractive investment for modernizing banking operations with minimal expenses.

Errors and its types

Finding and resolving bugs is essential to software development as it solves issues and ensures the program works properly.

Syntax Errors:

1. Ex-: Misspelled keywords, improper indentation, or missing colons.
2. Reason: Occur when the code violates syntax rules in the language, making compilation impossible (e.g., missing a colon after try or if).

Runtime Errors:

1. Ex-: Accessing a dictionary key that doesn't exist by dividing by zero.
2. Reason: Take place while the software is running. For instance, a mysql.connector may be triggered by your code containing incorrect database credentials.Did not succeed.

Value Errors:

1. Ex-: When a float is expected, a non-numeric value is entered.
2. Reason: Occurs when a function receives the correct type but the wrong value (insufficient initial deposit in create_account, for example).

Typing Errors:

1. An example is giving a string rather than a number.
2. Reason: Occur when an operation performs an incorrect type (e.g., entering non-float for the amount of the deposit).

Connection Errors:

1. Example: Database connection fails due to incorrect credentials or server issues.
2. Reason: Specific to network issues. In connect, wrong credentials or server downtime cause connection errors.

Operational Errors:

1. Ex-: SQL syntax errors or querying a non-existent table.
2. Reason: Result from database operation failures.

Testing

Functional Testing:

1. Verify that every feature works correctly using manually testing it.
2. Check the accuracy of opening, closing, depositing, and withdrawing accounts as well as checking balances.

Error Handling Testing:

1. Purposefully create errors (e.g., entering invalid inputs) and verify that the system handles them gently.
2. Check whether the system is showing the relevant error messages and continues running without crashing.

Database testing:

3. Verify that data is successfully stored and retrieved by analysing database connectivity and actions (such as CRUD operations).
4. Verify that accounts are created, updated, and deleted as expected.

Maintenance

1. Regular code review:

reviews on a regular basis to check for problems with coding standards.

2. Bug Fixes:

Promptly address reported bugs and test fixes to avoid regressions.

3. Database Maintenance:

For data protection, optimize your queries and make regular backups.

4. Security Updates:

Keep an eye out for security holes and install updates as soon as possible.

5. Documentation Maintenance:

For clarity, keep usage guidelines and updates current.

6. User Feedback:

Collect suggestions for improvements and rate them according to a requirement.

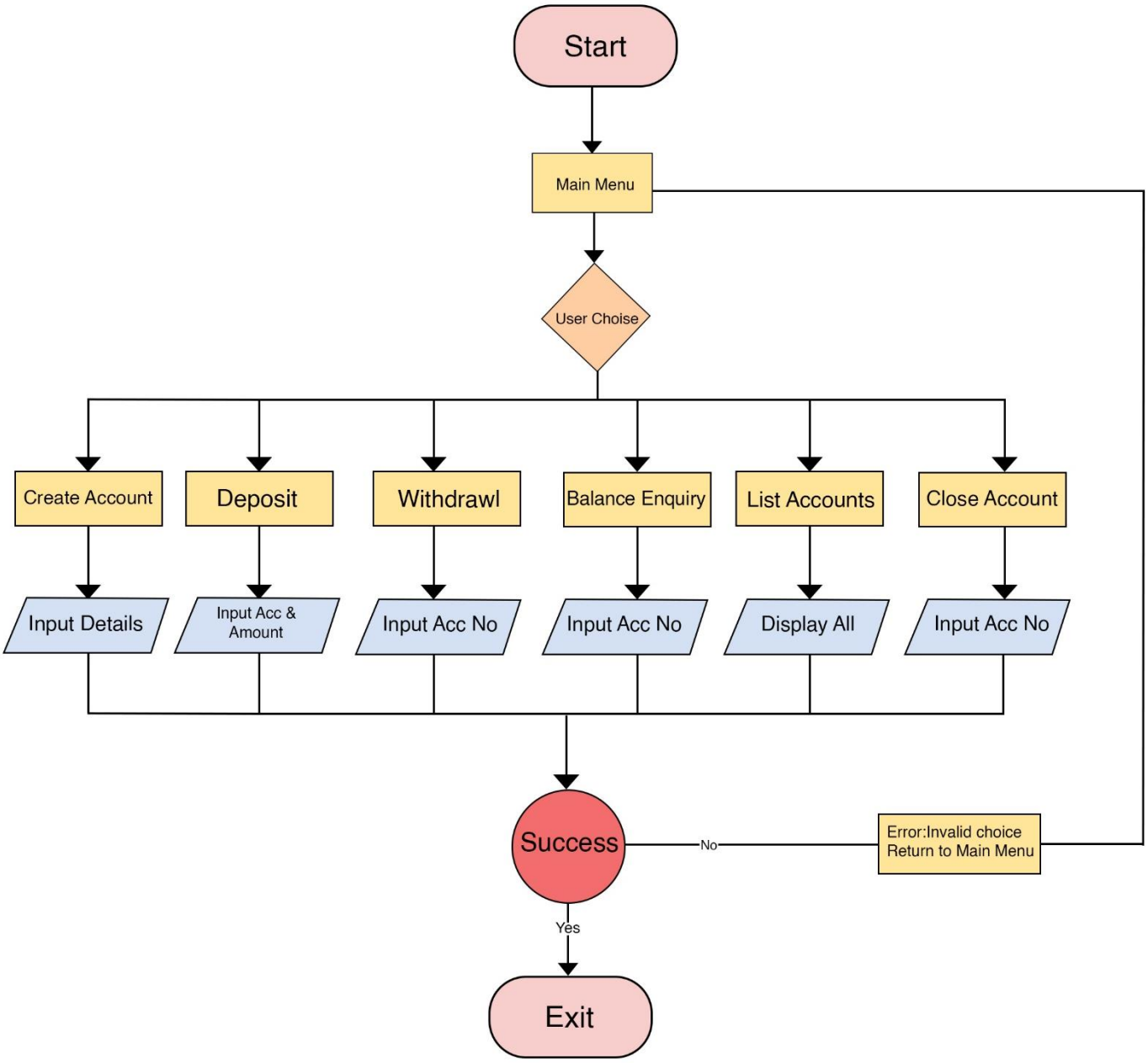
7. Version Control:

Effectively handle code modifications using version control systems.

8. Error Handling:

Keep an eye on logs for any recurring issues, and enhance error-handling mechanisms.

Flowchart


Start
Main Menu
User Choise
Create Account
Deposit
Withdrawl
Balance Enquiry
List Accounts
Close Account
Input Details
Input Acc & Amount
Input Acc No
Input Acc No
Display All
Input Acc No
Success
No
Error:Invalid choice
Return to Main Menu
Yes
Exit

User defined functions

1. connect():

This function establishes a connection to the MySQL database using the provided host, database name, user, and password. If the connection is successful, it returns a MySQL connection object; if not, it returns 'None'.

2. execute_query(query, *args, fetch_all=False):

The 'execute_query()' function executes a SQL query in the database. It takes the SQL query as the first parameter and optional parameters (*args) for the query values after that. The 'fetch_all' parameter determines whether to fetch all results or not. If 'fetch_all' is True, it returns the query result; if not, it returns 'None'.

3. create_account():

With the help of this function, users can open new bank accounts. The name of the account holder, the type of account (savings or current), and the amount of the initial deposit are required to be entered. If the account creation is successful, it creates a new record in the "accounts" database, issues a unique account number, and returns a success message. It raises a ValueError along with the relevant message if any input validation fails.

4. display_account(accNo):

Using the provided account number, the 'display_account()' function gathers and presents information about a particular account. If the account details are found, it queries the 'accounts' database and prints them; if not, it prints a failure message.

5. deposit_or_withdraw(accNo, amount, action):

With this feature, customers can add money to or take money out of a particular account. At first it checks the account's current balance, then updates it according to the action (deposit or withdraw) and amount that was mentioned. It prints a related success message if the transaction is successful and a failure message if it is not.

6. delete_account(accNo):

The 'delete_account()' function deletes a specified account from the 'accounts' table based on the provided account number. It prints a success message if the deletion is successful and a failure message if it is not.

7. display_all_accounts():

The 'accounts' table contains details about all the accounts that can be accessed and shown by this function. It prints the particulars of each account one by one after querying the table for all records. It prints a failure message if no accounts could be found.

Code

```
import mysql.connector
from mysql.connector import Error
import random

# Database connection details
DB_HOST = 'localhost'
DB_NAME = 'bankdb'
DB_USER = 'root'
DB_PASSWORD = 'anshika'


# Function to connect to the database
def connect():
    try:
        return mysql.connector.connect(
            host=DB_HOST,
            database=DB_NAME,
            user=DB_USER,
            password=DB_PASSWORD
        )
    except Error as e:
        print("Error connecting to the database.")
        return None


# Function to run SQL queries
def execute_query(query, *args, fetch_all=False):
    conn = connect()
    if conn is None:
        return None

    cursor = conn.cursor()
    try:
        cursor.execute(query, args)
        result = cursor.fetchall() if fetch_all else True
        conn.commit()
        return result
    except mysql.connector.Error as e:
        print("Error executing query:", e)
        return None
    finally:
        cursor.close()
        conn.close()


# Function to create a new account
def create_account():
    try:
```

```
        # Generate a new account number
        new_acc_no = '3366810000' + str(random.randint(1000,
9999))
        name = input("Enter the account holder's name:
").strip()
        acc_type = input("Enter the type of account (C/S):
").strip().upper()
        if acc_type not in ('C', 'S'):
            raise ValueError("Invalid input. Please enter C
for Current or S for Savings.")
        deposit = float(input("Enter the initial deposit
(>=500 for Savings and >=1000 for Current): "))
        if (acc_type == 'S' and deposit < 500) or (acc_type ==
'C' and deposit < 1000):
            raise ValueError("Invalid deposit amount.")

        query = "INSERT INTO accounts (accNo, name, accType,
deposit) VALUES (%s, %s, %s, %s)"
        if execute_query(query, new_acc_no, name, acc_type,
deposit):
            print(f"Account created successfully! Your account
number is {new_acc_no}")
    except ValueError as ve:
        print(str(ve))


# Function to display account details using last 4 digits
def display_account(last_4_digits):
    query = "SELECT * FROM accounts WHERE accNo LIKE %s"
    accounts = execute_query(query, f'%{last_4_digits}',
fetch_all=True)
    if accounts:
        for account in accounts:
            print(f"Account Number: {account[0]}, Account
Holder Name: {account[1]}, "
                  f"Type of Account: {account[2]}, Balance:
{account[3]}")
    else:
        print("No account found with the provided last 4
digits.")


# Function to deposit or withdraw money using the last 4
digits of the account number
def deposit_or_withdraw(last_4_digits, amount, action):
    query = "SELECT accNo, deposit FROM accounts WHERE accNo
LIKE %s"
    accounts = execute_query(query, f'%{last_4_digits}',
fetch_all=True)
    if not accounts:
        print("No account found with the provided last 4
digits.")
        return
```

```
    accNo = accounts[0][0]
    balance = accounts[0][1]

    if action == "deposit":
        query = "UPDATE accounts SET deposit = deposit + %s
WHERE accNo = %s"
        message = "Deposit"
    elif action == "withdraw" and balance >= amount:
        query = "UPDATE accounts SET deposit = deposit - %s
WHERE accNo = %s"
        message = "Withdrawal"
    else:
        print("Insufficient balance.")
        return

    if execute_query(query, amount, accNo):
        print(f"{message} successful!")
    else:
        print(f"Failed to {action} amount.")


# Function to delete an account using the last 4 digits
def delete_account():
    try:
        last_4_digits = input("Enter the last 4 digits of the
account number: ").strip()
        query = "SELECT accNo FROM accounts WHERE accNo LIKE
%s"
        accounts = execute_query(query, f'%{last_4_digits}',
fetch_all=True)
        if not accounts:
            print("No account found with the provided last 4
digits.")
            return

        accNo_to_delete = accounts[0][0]
        print(f"Deleting account with number:
{accNo_to_delete}")  # Debug print
        query = "DELETE FROM accounts WHERE accNo = %s"
        if execute_query(query, accNo_to_delete):
            print("Account deleted successfully!")
        else:
            print("Failed to delete the account.")
    except ValueError as ve:
        print("Invalid input.")


# Function to display all accounts
def display_all_accounts():
    query = "SELECT * FROM accounts"
    accounts = execute_query(query, fetch_all=True)
    if accounts:
        for account in accounts:
            print(f"Account Number: {account[0]}, Name:
```

```
{account[1]}, Type: {account[2]}, Balance: {account[3]}")
    else:
        print("Failed to fetch data.")


# Main function with menu
def main():
    while True:
        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. EXIT")
        ch = input("Select Your Option (1-7): ").strip()

        if ch == '1':
            create_account()
        elif ch == '2':
            last_4_digits = input("Enter the last 4 digits of
the account number: ").strip()
            amount = float(input("Enter the amount to deposit:
").strip())
            deposit_or_withdraw(last_4_digits, amount,
"deposit")
        elif ch == '3':
            last_4_digits = input("Enter the last 4 digits of
the account number: ").strip()
            amount = float(input("Enter the amount to
withdraw: ").strip())
            deposit_or_withdraw(last_4_digits, amount,
"withdraw")
        elif ch == '4':
            last_4_digits = input("Enter the last 4 digits of
the account number: ").strip()
            display_account(last_4_digits)
        elif ch == '5':
            display_all_accounts()
        elif ch == '6':
            delete_account()
        elif ch == '7':
            print("Thanks for using the bank management
system.")
            break
        else:
            print("Invalid choice. Please try again.")


if __name__ == "__main__":
    main()
```

Output

```
MAIN MENU
    1. NEW ACCOUNT
    2. DEPOSIT AMOUNT
    3. WITHDRAW AMOUNT
    4. BALANCE ENQUIRY
    5. ALL ACCOUNT HOLDER LIST
    6. CLOSE AN ACCOUNT
    7. EXIT
Select Your Option (1-7): 1
Enter the account holder's name: Aditya Kumar Gond
Enter the type of account (C/S): s
Enter the initial deposit (>=500 for Savings and >=1000 for Current): 500
Account created successfully! Your account number is 33668100002385
MAIN MENU
    1. NEW ACCOUNT
    2. DEPOSIT AMOUNT
    3. WITHDRAW AMOUNT
    4. BALANCE ENQUIRY
    5. ALL ACCOUNT HOLDER LIST
    6. CLOSE AN ACCOUNT
    7. EXIT
Select Your Option (1-7): 1
Enter the account holder's name: Anuj Roy
Enter the type of account (C/S): s
Enter the initial deposit (>=500 for Savings and >=1000 for Current): 600
Account created successfully! Your account number is 33668100006100
    MAIN MENU
    1. NEW ACCOUNT
    2. DEPOSIT AMOUNT
    3. WITHDRAW AMOUNT
    4. BALANCE ENQUIRY
    5. ALL ACCOUNT HOLDER LIST
    6. CLOSE AN ACCOUNT
    7. EXIT
Select Your Option (1-7): 2
Enter the last 4 digits of the account number: 2385
Enter the amount to deposit: 300
Deposit successful!
MAIN MENU
    1. NEW ACCOUNT
    2. DEPOSIT AMOUNT
    3. WITHDRAW AMOUNT
    4. BALANCE ENQUIRY
    5. ALL ACCOUNT HOLDER LIST
    6. CLOSE AN ACCOUNT
    7. EXIT
Select Your Option (1-7): 3
Enter the last 4 digits of the account number: 2385
Enter the amount to withdraw: 900
Insufficient balance.
MAIN MENU
    1. NEW ACCOUNT
    2. DEPOSIT AMOUNT
    3. WITHDRAW AMOUNT
    4. BALANCE ENQUIRY
    5. ALL ACCOUNT HOLDER LIST
    6. CLOSE AN ACCOUNT
```

```
    7. EXIT
Select Your Option (1-7): 4
Enter the last 4 digits of the account number: 2385
Account Number: 33668100002385, Account Holder Name: Aditya Kumar Gond,
Type of Account: S, Balance: 800.0
MAIN MENU
    1. NEW ACCOUNT
    2. DEPOSIT AMOUNT
    3. WITHDRAW AMOUNT
    4. BALANCE ENQUIRY
    5. ALL ACCOUNT HOLDER LIST
    6. CLOSE AN ACCOUNT
    7. EXIT
Select Your Option (1-7): 5
Account Number: 33668100002385, Name: Aditya Kumar Gond, Type: S, Balance:
800.0
Account Number: 33668100006100, Name: Anuj Roy, Type: S, Balance: 600.0
MAIN MENU
    1. NEW ACCOUNT
    2. DEPOSIT AMOUNT
    3. WITHDRAW AMOUNT
    4. BALANCE ENQUIRY
    5. ALL ACCOUNT HOLDER LIST
    6. CLOSE AN ACCOUNT
    7. EXIT
Select Your Option (1-7): 6
Enter the last 4 digits of the account number: 2385
Deleting account with number: 33668100002385
Account deleted successfully!
    MAIN MENU
    1. NEW ACCOUNT
    2. DEPOSIT AMOUNT
    3. WITHDRAW AMOUNT
    4. BALANCE ENQUIRY
    5. ALL ACCOUNT HOLDER LIST
    6. CLOSE AN ACCOUNT
    7. EXIT
Select Your Option (1-7): 7
Thanks for using the bank management system.
```

Conclusion

Using Python and MySQL, the Bank Management System project was able to solve the essential problems related to account management. Careful debugging and teamwork overcame difficulties like syntax errors and runtime problems to make the project effective. We are appreciative of Mrs. Tanusree Sadhukhan's guidance and the resources offered by PM Shri Kendriya Vidyalaya Panagarh. In the future, this technology will increase efficiency and customer service by giving banking operations a solid base.

Bibliography

For successfully completing my project file. I have taken help from the following websites links:-

www.google.com

www.youtube.com

Discover the code that powers this project! Gain access to the source code on my GitHub repository:

soft-ad/bank management system at main · Face51/soft-ad (github.com)

Thankyou