INTRODUCTION

Phoenix bank management system is a secure platform offering various banking operations. Upon launching, you'll encounter the main menu with options ranging from creating a new account to managing existing ones, conducting deposits and withdrawals, transferring funds, accessing the manager's menu for additional functionalities, deleting accounts, and finally, exiting the application.

AIM

The primary aim of this project is to provide students with a practical application of their programming skills in real-world scenarios, enhancing their understanding of how programming contributes to robust software development.

Key objectives include:

1. Proficiency in writing programs using contemporary software tools.
2. Effective application of object-oriented programming principles in the development of small to medium-sized projects.
3. Mastery of writing procedural code to address small to medium-sized problems.
4. Demonstration of comprehensive knowledge in informatics practices, encompassing systems, theory, and software development.
5. Showcase students' ability to conduct research or applied informatics practices projects, requiring strong writing and presentation skills in a scholarly style.

DISCRIPTION OF EXISTING SYSTEM

In the existing system, the records are maintained manually and the paper Work is more.

- Entering Record

Entry of each record is done manually and each time the record is maintained on paper, it maximizes the maintenance of additional files.

- Searching the record

Due to absence of unique identification, a person has to search the record and this resulted in increased wastage of time.

- Sorting of Records

All the records of Account's is maintained on papers and if in any case we want to see any particular record, we have to search many pages.

DISCRIPTION OF PURPOSED SYSTEM

The proposed system avoids the limitation of current system and has the following benefit over the existing System.

• Everything is automated which reduce the risk factor.

• Flexibility in generating of information.

• Quick retrieval and ease of maintenance of data.

• Highly accurate.

• User satisfaction.

PROGRAMMING LANGUAGE USED

Python: Python is a general-purpose, dynamic, high-level, and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides many high-level data structures.

Guido Van Rossum is the founder of Python programming language.

Features of Python:

- Python is a high-level language. It is a free and open-source language.

- It is an interpreted language, as Python programs are executed by an interpreter.

- Python programs are easy to understand as they have a clearly defined syntax and relatively simple structure.

- Python is case-sensitive.

- Python is portable and platform independent, means it can run on various operating systems and hardware platforms.

- Python has a rich library of predefined functions.

- Python is also helpful in web development. Many popular web services and applications are built using Python, like Instagram, YouTube, Uber, Facebook, Netflix etc.

- Python uses indentation for blocks and nested blocks.

IDE Used

Spyder:

Spyder is a free and open-source Python scientific environment for scientists, Engineers and data analysts. Combining advanced coding, analysis, debugging, and profiling tools, it's designed for scientific data exploration and visualization.

VSCode:

VSCode, a developer-focused code editor, offers powerful coding, analysis, debugging, and profiling tools, mirroring an integrated development environment. Built on Visual Studio Code's open-source base, it caters to programmers' needs.

Python Libraries Used

MySQL-connector:

The purpose of this plugin is to connect with MySQL database and perform query and insert and store data in organized tables.

PyInputPlus:

Provides more featureful versions of input () and raw input (), it basically accepts only user defined data type, to avoid random errors.

Numpy:

NumPy is used for numerical computing, offering powerful support for multi-dimensional arrays and mathematical functions. It simplifies complex operations and facilitates efficient handling of large datasets.

Shutil:

Shutil is a module, used to simplify common file-related tasks and provides a convenient way to interact with the file system. We have use shutil to retrieve the terminal/console width for printing even lines

Random:

Random is a module that provides functions for generating pseudo-random numbers. We have used random to randomize the account number for each new customer

PyFiglet:

PyFiglet is a wrapper for the FIGlet program, which is used for creating text banners in various typefaces composed of letters made up of conglomerations of smaller ASCII characters. FIGlet is a popular tool for creating stylized text art and ASCII banners.
We have used this purely for aesthetic purpose

Rich:

Rich is used for designing and enhancing the visual appearance of terminal text output. It allows adding colors, styles, and formatting options to text displayed in the console. With features such as text styling, table creation with customizable styles, syntax highlighting, and progress bars, it provides a toolkit for creating more visually appealing and readable command-line interfaces.

Colorama:

Colorama simplifies the process of adding colored output to terminal text. It provides an easy way to add colored foreground and background text to print statements, making it visually appealing and enhancing the readability of text displayed in the console.

Tabulate:

Tabulate facilitates the creation of formatted tables in console applications. It takes a list of dictionaries or other tabular data structures and formats them into a visually appealing table.

Sys:

Sys module provides access to some variables used or maintained by the Python interpreter and functions that interact strongly with the interpreter. It is often used for system-specific configurations and settings. We used sys.exit to exit the system

Os:

Os module provides a way of interacting with the operating system. It includes functions to perform operating system-dependent operations, such as reading or writing to the file system, working with directories, and executing commands. We have used it to figure out if it is running in windows or UNIX based systems

Backend (To Store Data) Used

SQL:

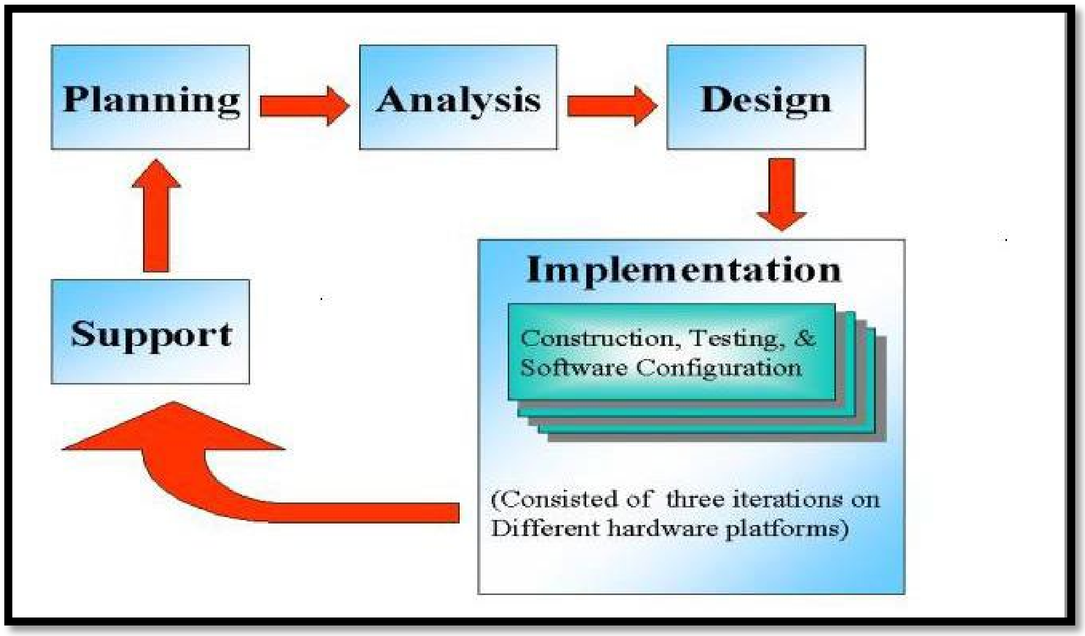
Structured Query Language (SQL) is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data, i.e., data incorporating relations among entities and variables.

Backend Software used

MySQL:

MySQL is an open-source relational database management system (RDBMS) that enables efficient storage, retrieval, and management of data. It's known for its robustness, scalability, and speed. MySQL is widely used for various applications, from web development to business solutions, making it a popular choice in the world of databases.

SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)


Planning
Analysis
Design
Implementation
Construction, Testing, & Software Configuration
(Consisted of three iterations on Different hardware platforms)
Support

The systems development life cycle is a project management method that breaks complex projects into smaller, manageable phases.

This helps ensure that each phase is successful before moving on to the next one.

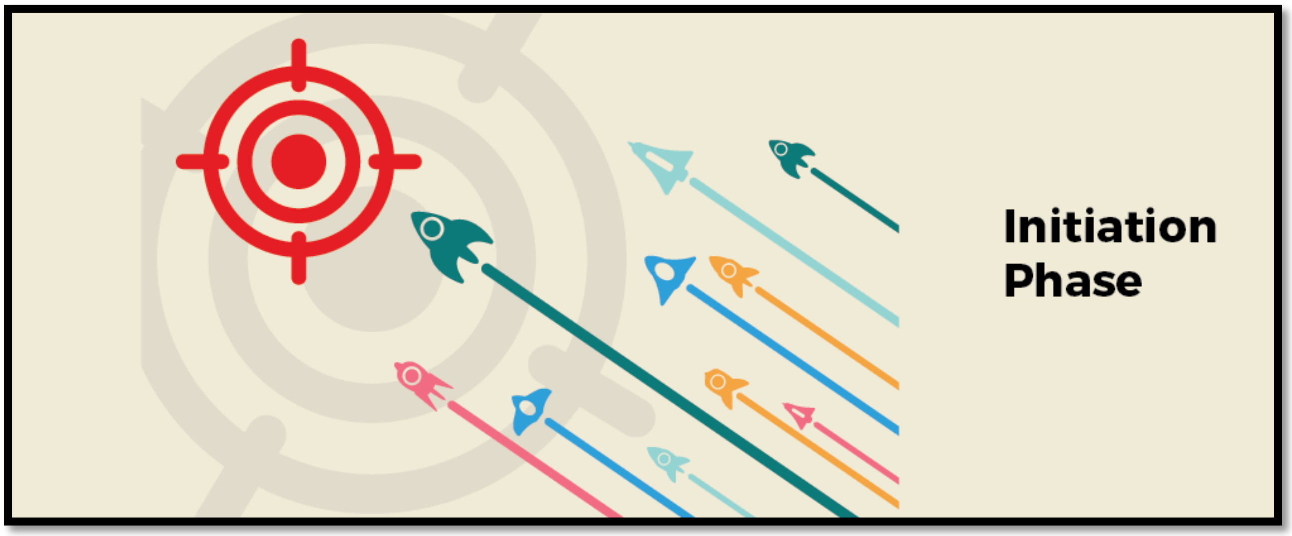
Software development projects generally include stages like:

- Initiation
- Planning
- Design
- Development
- Testing
- Implementation
- Maintenance.

However, organizations may divide them differently. For instance, initial project tasks might be called request, requirements definition, and planning phases. Involving end users in each phase is crucial to ensure the system meets their needs.

PHASES OF SYSTEM DEVELOPMENT LIFE CYCLE

INITIATION PHASE

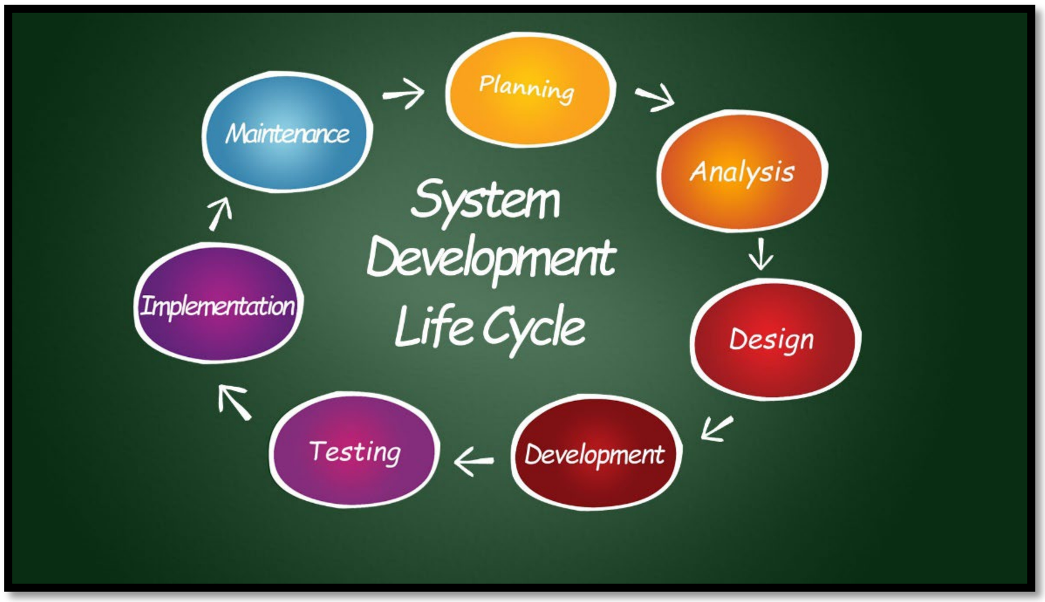


The Initiation Phase begins when a business sponsor identifies a need or an opportunity. The purpose of the Initiation Phase is to:

1. Identify and validate opportunities for enhancing business achievements or addressing specific needs.
2. Clarify assumptions and constraints for potential solutions.
3. Explore alternative concepts and methods, including process changes.
4. Secure support from executive leaders via a Concept Proposal.
5. Align the project with organizational infrastructure and plans, resulting in a Project Management Charter for the project manager's authority.

Effective oversight ensures project alignment with strategic goals. The initiation phase identifies improvement opportunities through a concise business case, outlining purpose, benefits, and requirements.

SYSTEM CONCEPT DEVELOPMENT PHASE



The System Concept Development Phase begins after a business need or the Agency/Organization Program Leadership and the Agency/Organization CIO validate opportunity. In the System Concept Development Phase,

We undertake the following key steps:

1. Assess the practicality of our ideas.
2. Establish connections with other systems.
3. Define system functionality and data requirements.
4. Set clear success metrics and objectives.
5. Evaluate cost-effectiveness and benefits of various development approaches.
6. Identify and address potential challenges.
7. Develop the primary technical plan.
8. Make decisions about software choices and release strategies.
9. Test simplified versions for technology validation.
10. Leverage the System Boundary Document to secure State CIO approval for our project.

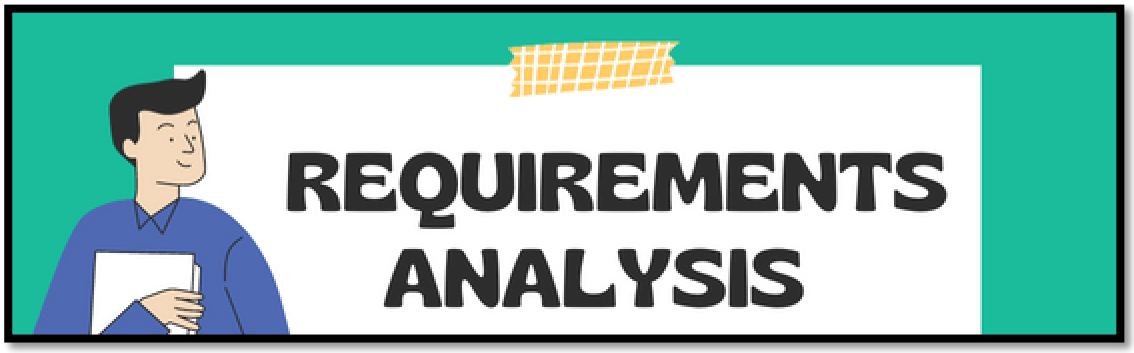
PLANNING PHASE

The planning phase is crucial for all projects, ensuring effective coordination and risk management. Project plans should match project characteristics and risks.

They refine the initiation phase's information by detailing activities and resources. Project managers play a vital role in coordinating discussions among various teams to identify functional, security, and network requirements.

This phase creates a plan with methods, tools, tasks, resources, schedules, and user input. It also establishes personnel assignments, costs, and target dates. A Project Management Plan covers acquisition, configuration management, quality assurance, operations, security, verification, validation, and systems engineering management.

REQUIREMENTS AND ANALYSIS PHASE



This phase is all about getting into the nitty-gritty details. We take the high-level requirements from earlier phases and make them very specific. This includes defining things like data needs, system performance, security, and how easy it is to maintain the system.

These requirements need to be clear and testable. They should be directly related to the business needs we identified in the beginning. We will use these requirements to decide if the system is working correctly, which we will outline in the Test and Evaluation Master Plan.

In this phase, we:

1. Get into the specifics of functional and data requirements and put them in a Requirements Document.
2. Check and improve how our business processes work, including what data drives them and who handles it.
3. Create detailed models of data and processes, such as what goes into the system and what comes out.
4. Plan how we'll test the system to make sure it works as it should.

DESIGN PHASE

In the design phase, we turn the requirements from earlier phases into detailed plans that developers use to create the software. There are two main ways we do this:

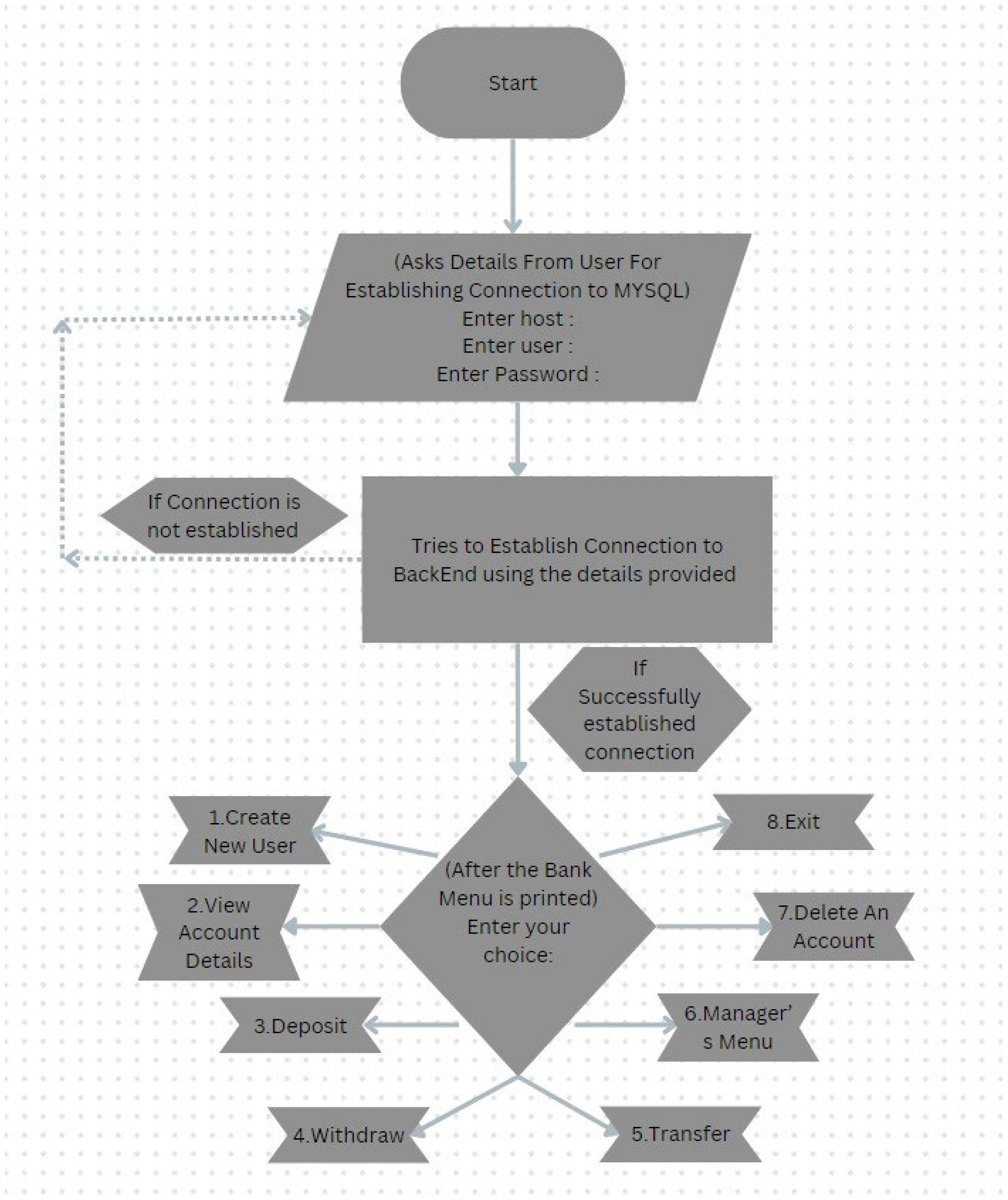
1. Top-down: We start with big program pieces and how they connect, then break it down into smaller parts.
2. Bottom-up: We begin with small program parts and connections, and then build up to larger systems.

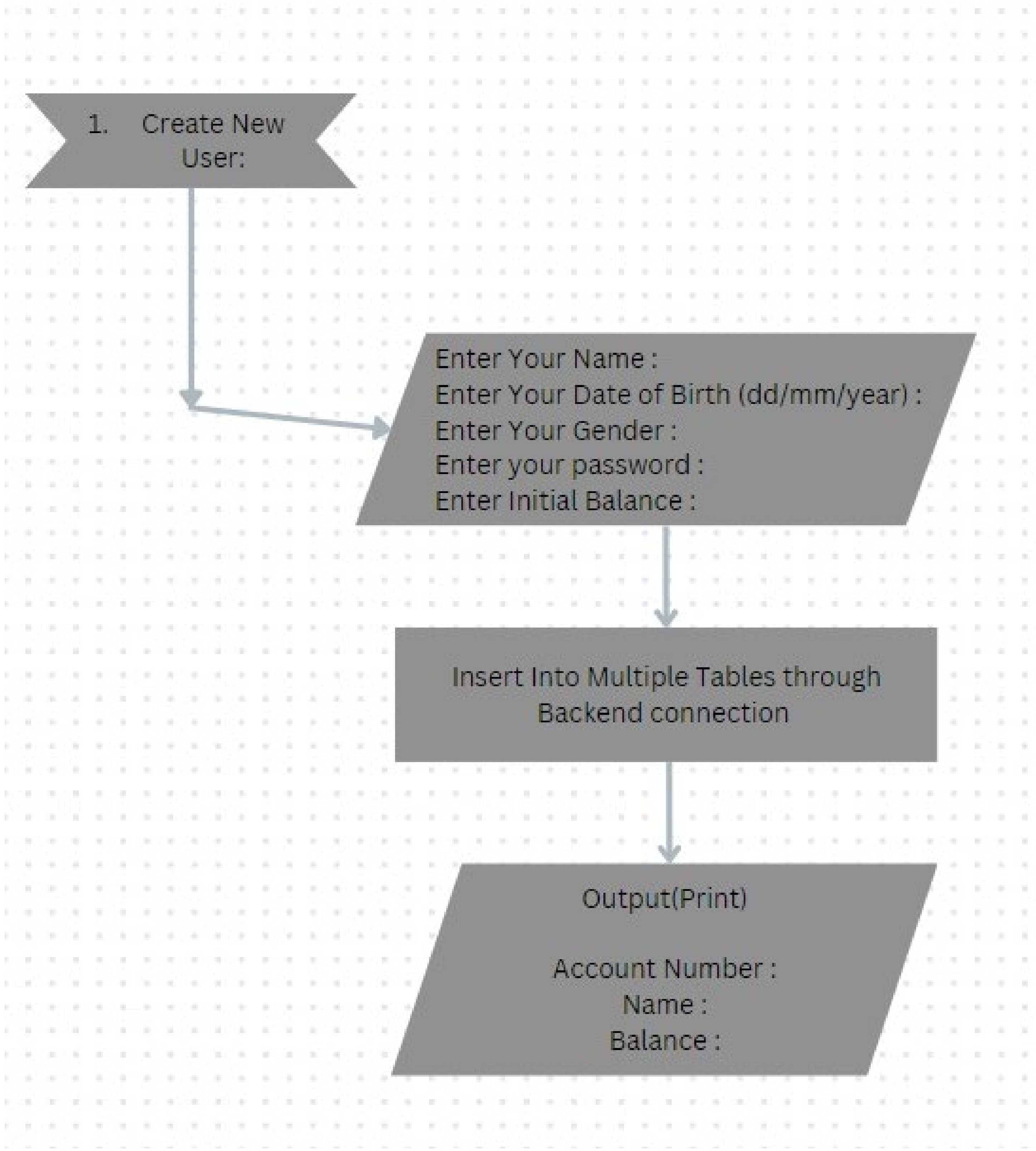
We often use prototyping tools to create mock-up designs of things like screens and databases. Different teams review and refine these designs until they are approved. During this phase, we design the system to meet the functional requirements. To avoid costly problems later, we identify and address potential issues, including:

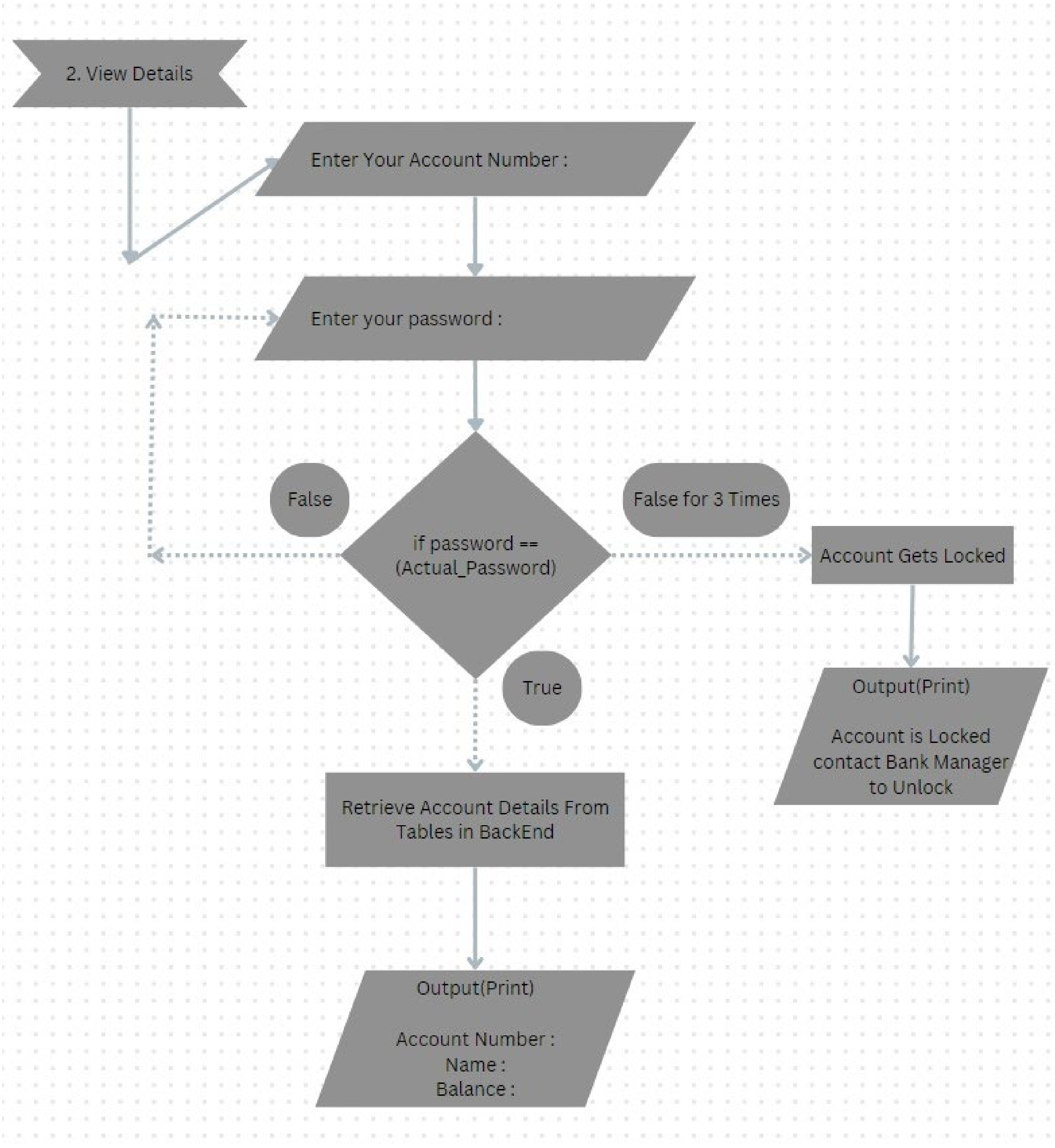
- Risks and how to deal with them.
- Security risks.
- Moving data from the old system to the new one.
- The system's environment.
- Subsystems and how they work together.
- Allocating tasks to resources.
- Detailed specifications for each software part.

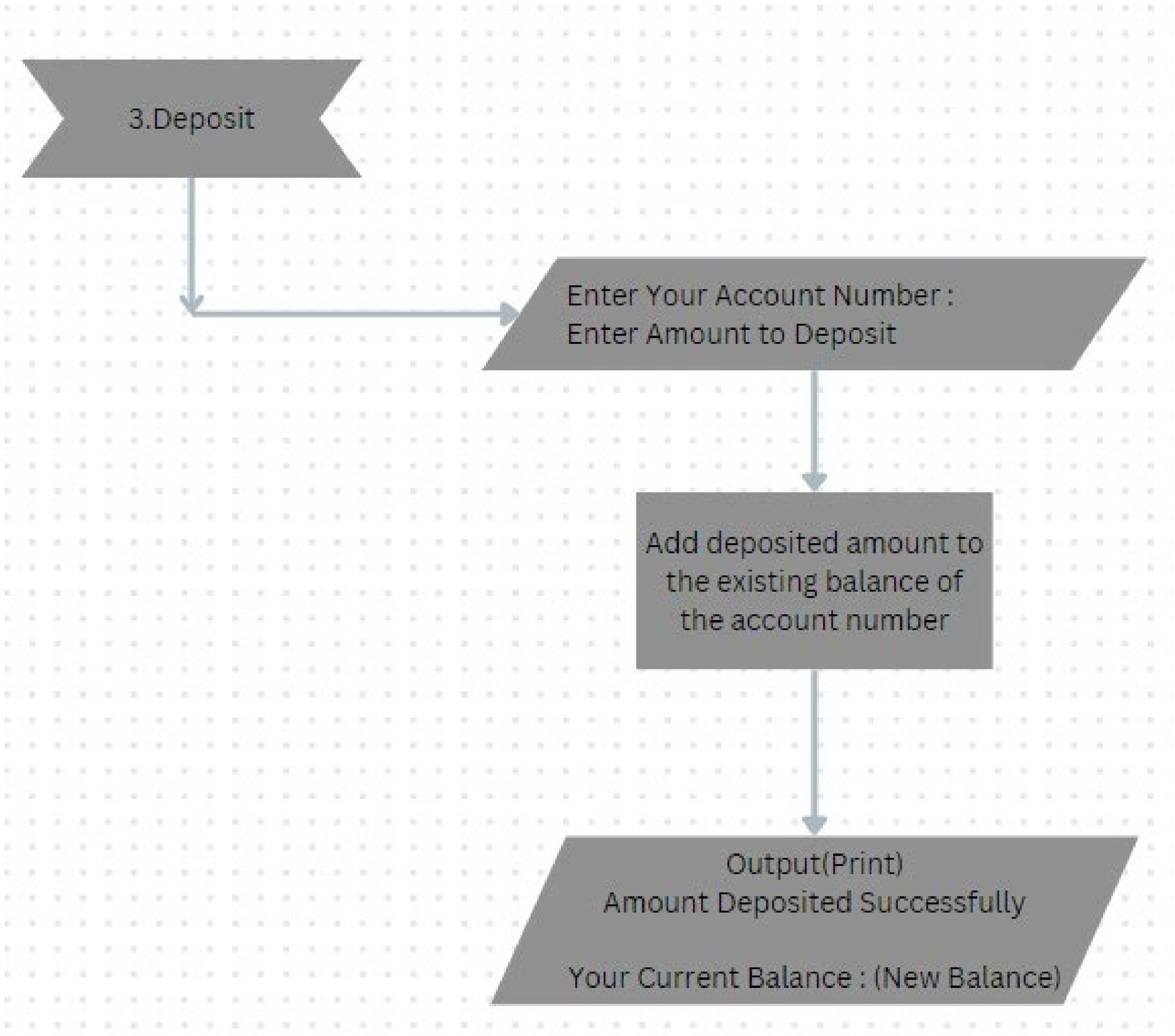
This results in an initial System Design Document, followed by a comprehensive review for approval. Simultaneously, we initiate plans for system implementation, operation, maintenance, and training.

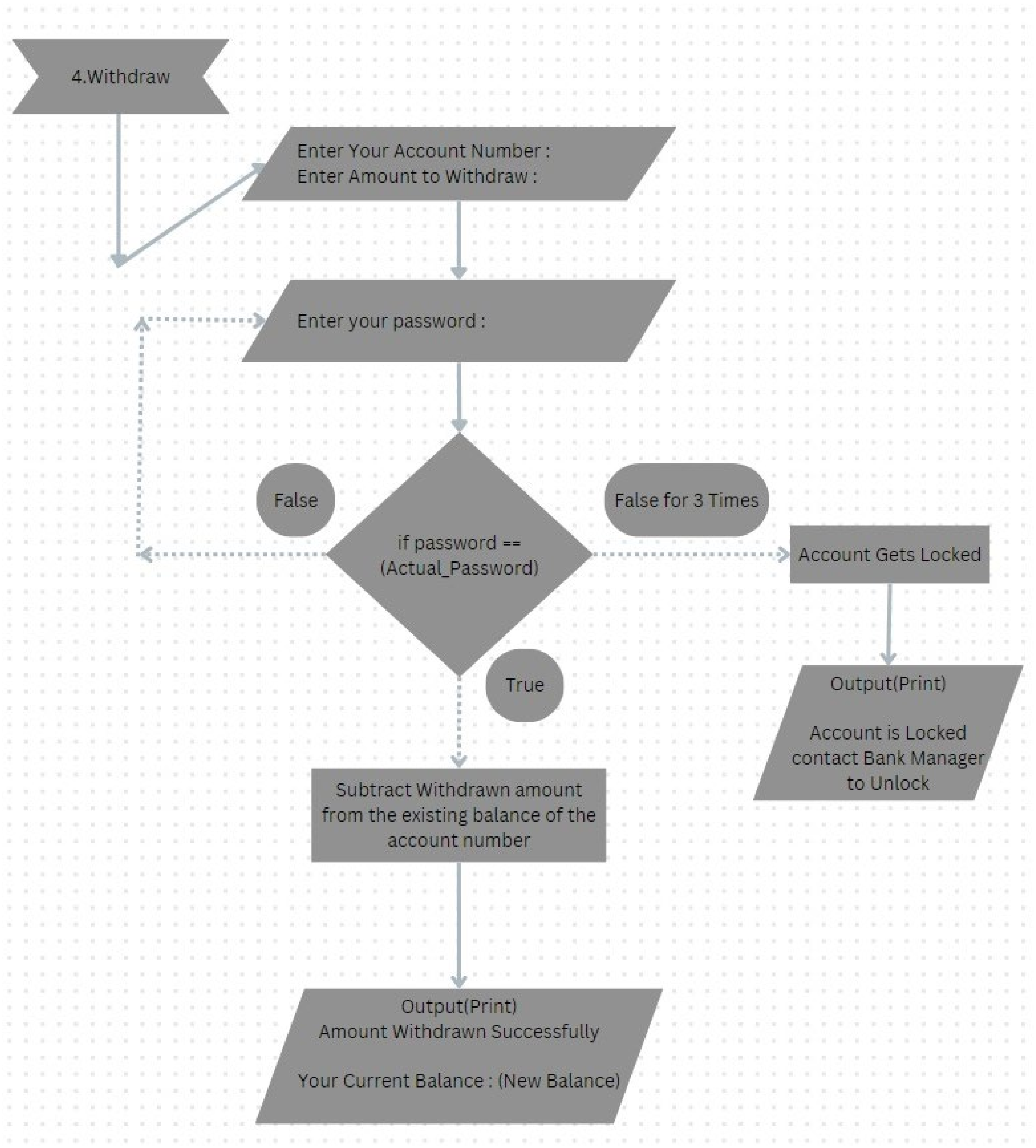
Flow Chart

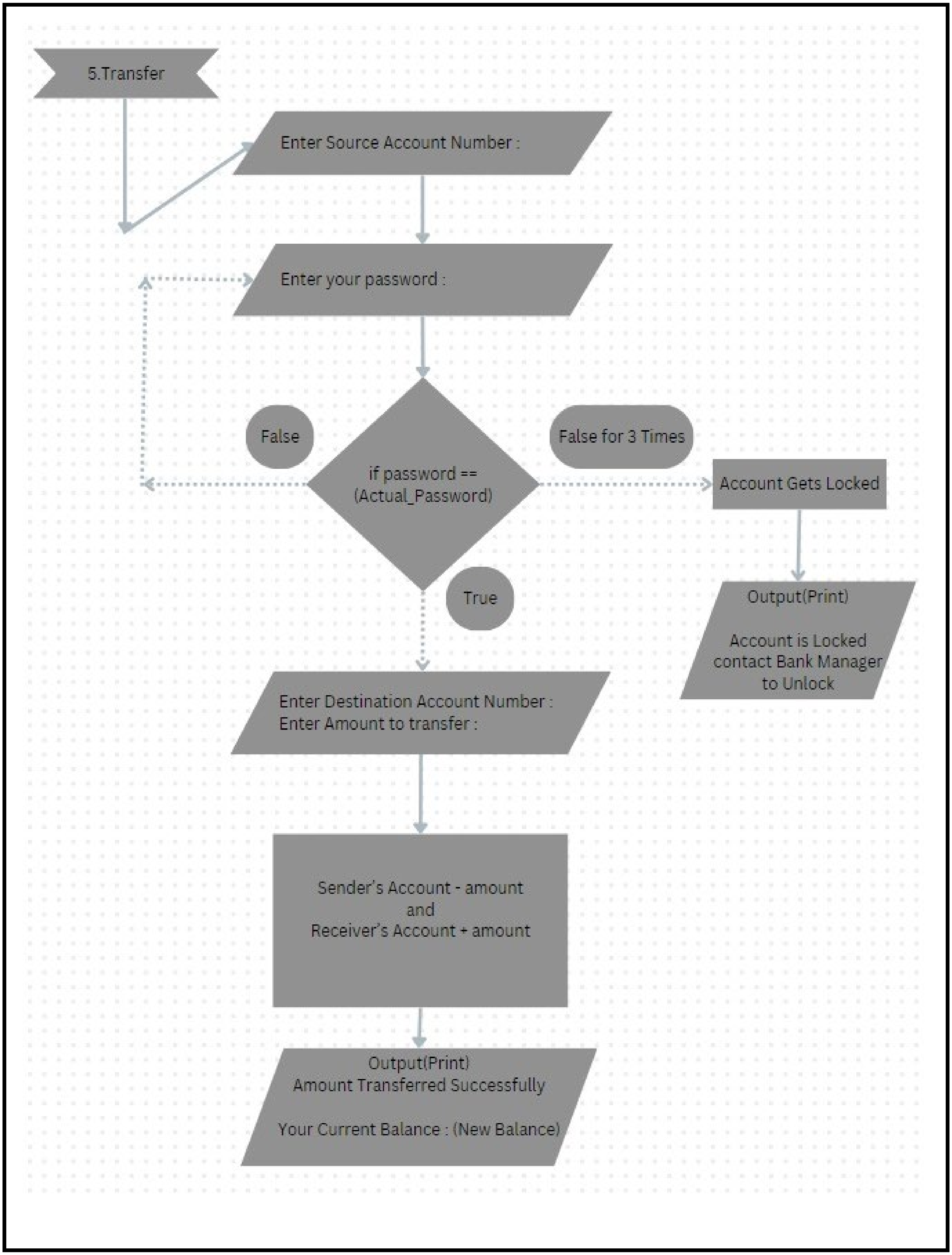

Start
(Asks Details From User For Establishing Connection to MYSQL)
Enter host :
Enter user :
Enter Password :
If Connection is not established
Tries to Establish Connection to BackEnd using the details provided
If Successfully established connection
(After the Bank Menu is printed) Enter your choice:
1.Create New User
2.View Account Details
3.Deposit
4.Withdraw
5.Transfer
6.Manager's Menu
7.Delete An Account
8.Exit

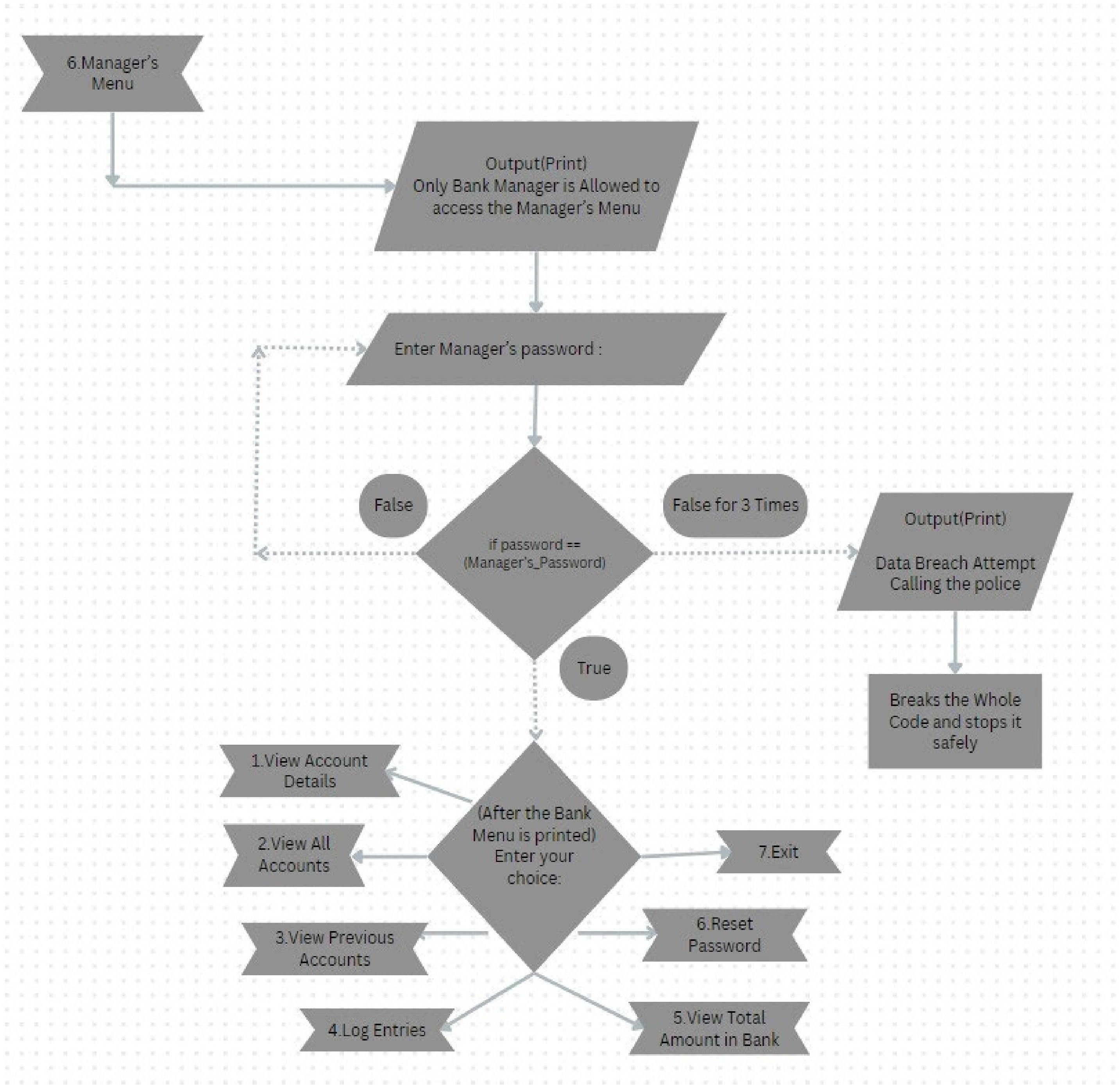

1. Create New User:
Enter Your Name :
Enter Your Date of Birth (dd/mm/year) :
Enter Your Gender :
Enter your password :
Enter Initial Balance :
Insert Into Multiple Tables through Backend connection
Output(Print)
Account Number :
Name :
Balance :

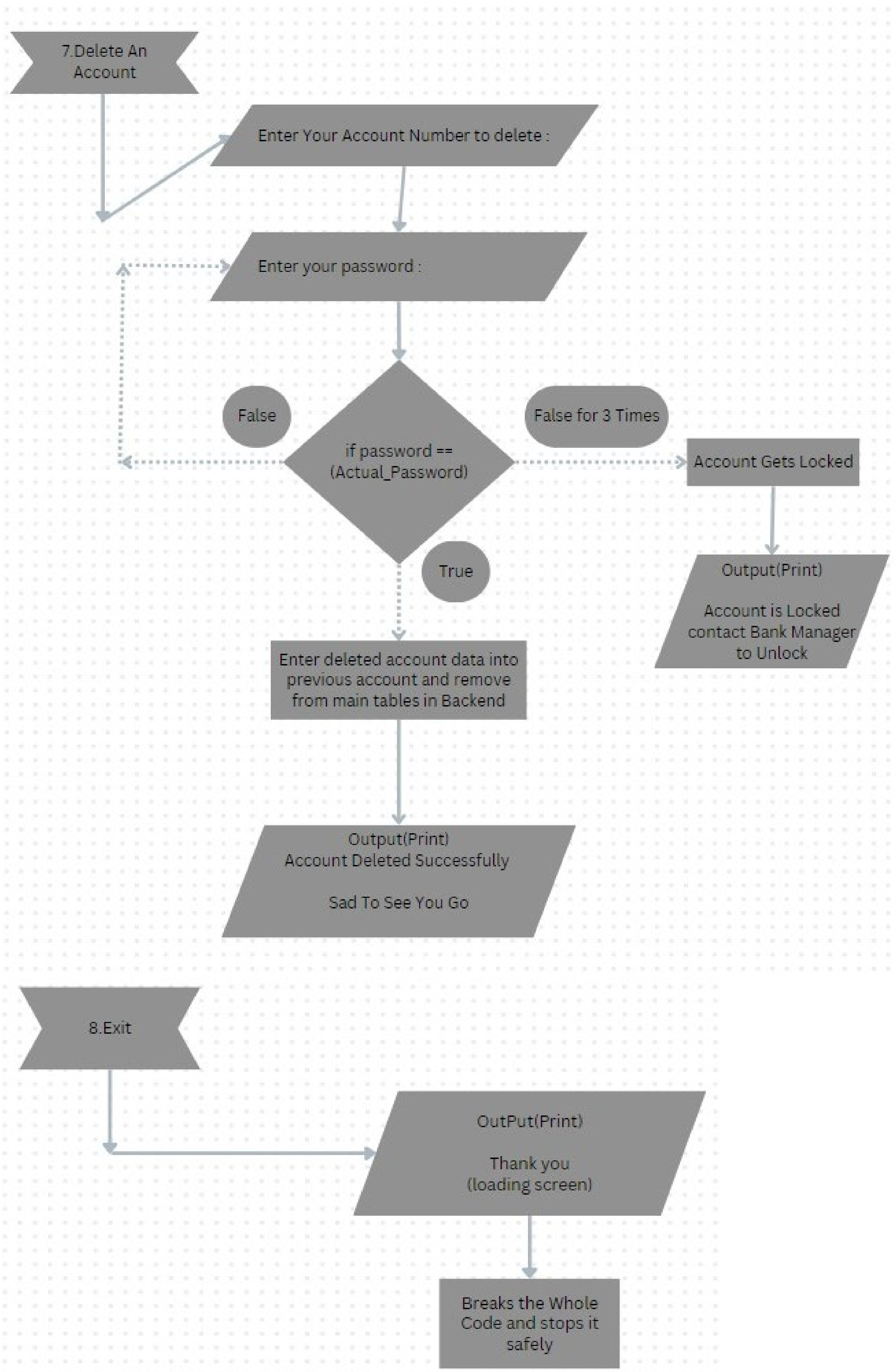

2. View Details
Enter Your Account Number :
Enter your password :
False
if password == (Actual_Password)
False for 3 Times
Account Gets Locked
True
Output(Print)
Account is Locked contact Bank Manager to Unlock
Retrieve Account Details From Tables in BackEnd
Output(Print)
Account Number :
Name :
Balance :


3.Deposit
Enter Your Account Number :
Enter Amount to Deposit
Add deposited amount to the existing balance of the account number
Output(Print)
Amount Deposited Successfully
Your Current Balance : (New Balance)

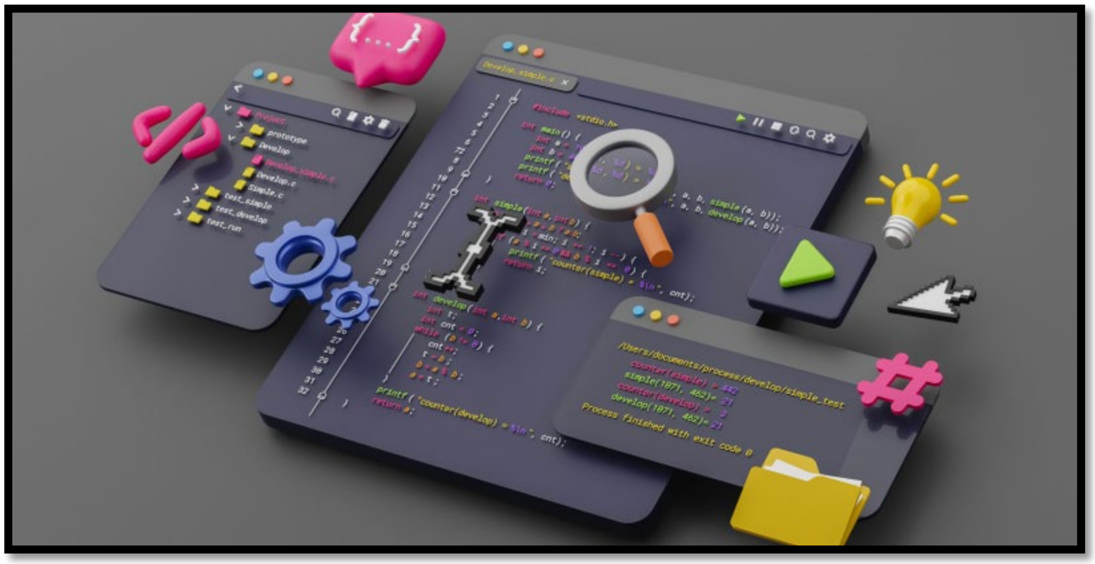

4.Withdraw
Enter Your Account Number :
Enter Amount to Withdraw :
Enter your password :
False
if password == (Actual_Password)
False for 3 Times
Account Gets Locked
True
Output(Print)
Account is Locked contact Bank Manager to Unlock
Subtract Withdrawn amount from the existing balance of the account number
Output(Print)
Amount Withdrawn Successfully
Your Current Balance : (New Balance)


5.Transfer
Enter Source Account Number :
Enter your password :
False
if password == (Actual_Password)
False for 3 Times
Account Gets Locked
True
Output(Print)
Account is Locked contact Bank Manager to Unlock
Enter Destination Account Number :
Enter Amount to transfer :
Sender's Account - amount
and
Receiver's Account + amount
Output(Print)
Amount Transferred Successfully
Your Current Balance : (New Balance)


6.Manager's Menu
Output(Print) Only Bank Manager is Allowed to access the Manager's Menu
Enter Manager's password :
False
if password == (Manager's_Password)
False for 3 Times
Output(Print) Data Breach Attempt Calling the police
Breaks the Whole Code and stops it safely
True
(After the Bank Menu is printed) Enter your choice:
1.View Account Details
2.View All Accounts
3.View Previous Accounts
4.Log Entries
5.View Total Amount in Bank
6.Reset Password
7.Exit


7.Delete An Account
Enter Your Account Number to delete :
Enter your password :
False
if password == (Actual_Password)
False for 3 Times
Account Gets Locked
Output(Print)
Account is Locked contact Bank Manager to Unlock
True
Enter deleted account data into previous account and remove from main tables in Backend
Output(Print)
Account Deleted Successfully
Sad To See You Go
8.Exit
OutPut(Print)
Thank you
(loading screen)
Breaks the Whole Code and stops it safely

DEVELOPMENT PHASE



During the development phase, we take the detailed plans from the design stage and turn them into actual computer programs. To make this process smooth, it is crucial for programmers and other project members to talk about the design specifications before they start writing code.

This ensures that everyone understands how the program should work and what it needs to achieve. Programmers use different methods to create these computer programs. For example, in some cases, especially with large financial systems, they use a method called "procedural programming".

This means they write out logical instructions systematically, like putting together a jigsaw puzzle to form a complete program.

In this development phase, we do the following:

1. We take the detailed requirements and designs and use them to create the different parts of the system.
2. We test each part on its own to make sure it works as intended.
3. We get ready to bring all these parts together and test the entire IT system to make sure it functions correctly as a whole.

Source Code

```
# ────── IMPORT MODULES──────────────────────────────────────────────────────────

import os, sys, time, shutil, msvcrt, random, pyfiglet

import numpy as np, pandas as pd, pyinputplus as pyip, mysql.connector as sql

from pyfiglet import Figlet
from datetime import datetime
from tabulate import tabulate
from colorama import Fore, Style

from rich import box
from rich.text import Text
from rich.panel import Panel
from rich.table import Table
from rich.progress import track
from rich.console import Console

console = Console()
Print = console.print

# ────── PRE-REQUISITE(Security) ────────────────────────────────────────────────

# HIDES THE TYPED PASSWORD WITH (*) SO IT IS NOT VISIBLE TO ANYONE
def masked_input(prompt=""):
    if os.name == 'nt':
        # Windows platform
        print(prompt, end='', flush=True)
        password = []
        while True:
            char = msvcrt.getch()
            if char == b'\r':  # Enter key pressed
                print()
                break
            elif char == b'\x08':  # Backspace key pressed
                if password:
                    password.pop()  # Remove the last character
                    sys.stdout.write('\b \b')  # Clear the character on the
screen
                    sys.stdout.flush()
            else:
                password.append(char.decode('utf-8'))
                sys.stdout.write('*')  # Display asterisks instead of characters
                sys.stdout.flush()
        return ''.join(password)
    else:
        # Unix-Like platforms
        print(prompt, end='', flush=True)
        password = []
        while True:
            char = sys.stdin.read(1)

            if char == '\n':  # Enter key pressed
                print()
                break
            elif char == '\x08':  # Backspace key pressed
                if password:
                    password.pop()  # Remove the last character
                    sys.stdout.write('\b \b')  # Clear the character on the
screen
                    sys.stdout.flush()
            else:
                password.append(char)
                sys.stdout.write('*')  # Display asterisks instead of characters
                sys.stdout.flush()
        return ''.join(password)

# A PRETTY DECORATIVE HORIZONTAL LINE
def print_horizontal_line():
    terminal_width = shutil.get_terminal_size().columns
    horizontal_line = "—" * terminal_width
    print(horizontal_line)

# ——————— LINK TO MYSQL (BACKEND) ————————————————————————————————————————————————

# RESQUEST'S THE USER TO INPUT CONNECTION DETAILS FOR ESTABLISHING BACKEND
CONNECTION
host = masked_input("Enter Host:")
user = masked_input("Enter User:")
password = masked_input("Enter Password:")

# CONVERTS THE CONNECTION DETAILS TO STRING FORMAT
host = str(host)
user = str(user)
password = str(password)

# ESTABLISHE'S CONNECTION TO BACKEND (MySQL)
conn = sql.connect(host = 'localhost',
                   user = 'root',
                   password = '1234567890')
cursor = conn.cursor()

# CREATE'S THE DATABASE NAMED (BANK) IF IT DOESN'T EXISTS
cursor.execute("CREATE DATABASE IF NOT EXISTS bank")
cursor.close()

# ESTABLISHES CONNECTION DIRECTLY TO BANK DATABASE
idpass = sql.connect(host = 'localhost',
                   user = 'root',
                   password = '1234567890',
                     database='bank')
cursor = idpass.cursor()

Print(Panel("[aquamarine1]Connection Success!"), justify = "center")
print_horizontal_line()
# ────── TO CREATE TABLES IN DATABASE────────────────────────────────────────────

# TO RECORD THE ACCOUNT NUMBER, BALANCE AND STATUS
def create_account():
    query = '''CREATE TABLE IF NOT EXISTS account (
        acc_number INT PRIMARY KEY,
        balance FLOAT NOT NULL,
        status TINYINT(1) NOT NULL DEFAULT 0)'''
    cursor.execute(query)
    idpass.commit()

# SEPERATE TABLE MAINTAINED FOR PASSWORDS FOR SECURITY REASONS
def create_password():
    query = '''CREATE TABLE IF NOT EXISTS password (
        acc_number INT PRIMARY KEY,
        passcode VARCHAR(30) NOT NULL)'''
    cursor.execute(query)
    idpass.commit()

# STORES CUSTOMERS PERSONAL INFORMATION (DOB, Name, Gender)
def create_customer_detail():
    query = '''CREATE TABLE IF NOT EXISTS customer_detail (
        acc_number INT PRIMARY KEY,
        dob DATE NOT NULL,
        name VARCHAR(30) NOT NULL,
        gender VARCHAR(11) NOT NULL)'''
    cursor.execute(query)
    idpass.commit()

# TO RECORDS ALL THE TRANSACTIONS AND ACTIONS OCCURED ALONG WITH THE
# (id, ToA/Time of (action/transaction), acc_num, type of transaction, amount
involved)
def create_log():
    query = '''CREATE TABLE IF NOT EXISTS log (
        id INT AUTO_INCREMENT PRIMARY KEY,
        toa TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        acc_number INT NOT NULL,
        transaction_type VARCHAR(50) NOT NULL,
        amount FLOAT)'''
    cursor.execute(query)
    idpass.commit()

# RECORDS IF ANYONE DONATED TO CHARITY WHILE DEPOSITING OR WITHDRAWING
def create_charity():
    query = '''CREATE TABLE IF NOT EXISTS charity(
        transaction_id INT,
        donated_time TIMESTAMP,
        acc_number INT)'''
    cursor.execute(query)
    idpass.commit()

# STORES THE DELETED ACCOUNTS AND RECORDS ALL THE DETAILS IN THE SAME TABLE
def create_past_customer():
    query = '''CREATE TABLE IF NOT EXISTS past_customer (
        deleted_on TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        acc_number INT PRIMARY KEY,
        dob DATE NOT NULL,
        name VARCHAR(30) NOT NULL,
        gender VARCHAR(11) NOT NULL)'''
    cursor.execute(query)
    idpass.commit()

# ——— CREATING A NEW ACCOUNT ————————————————————————————————

# KEEPS RECORD OF GENERATED ACCOUNT NUMBER TO NEVER REPEAT THEM AGAIN
generated_numbers = set()

# IT ALWAYS GENERATES RANDOM 6 DIGIT ACCOUNT NUMBERS
def generate_random_number():
    while True:
        random_number = random.randint(100000, 999999)
        if random_number not in generated_numbers:
            generated_numbers.add(random_number)
            return random_number

random_number = generate_random_number()

# CREATES AND RECORDS THE NEW ACCOUNTS INTO ALL THE REQUIRED TABLES
def create_new_account(acc_num, name, dob, gender, balance, passcode):

    query = '''INSERT INTO account (acc_number, balance)
    VALUES (%s, %s)'''
    values = (acc_num, balance)
    cursor.execute(query, values)
    idpass.commit()

    query = '''INSERT INTO password (acc_number, passcode)
    VALUES (%s, %s)'''
    values = (acc_num, passcode)
    cursor.execute(query, values)
    idpass.commit()

    dobh = dob.split('/')
    day = int(dobh[0])
    month = int(dobh[1])
    year = int(dobh[2])
    dob_value = datetime(year, month, day).strftime('%Y-%m-%d')

    query = '''INSERT INTO customer_detail  (acc_number, dob, name, gender)
    VALUES (%s, %s, %s, %s)'''
    values = (acc_num, dob_value, name, gender)
    cursor.execute(query, values)
    idpass.commit()

    query = '''INSERT INTO log (acc_number, transaction_type, amount)
    VALUES (%s, %s, %s)'''
    values = (acc_num, 'created account', balance)
    cursor.execute(query, values)
    idpass.commit()

# ────── SECURITY ──────────────────────────────────────────────────

# CHECKS THE MINIMUM REQUIRMENT FOR PASSSWORDS
# i.e. length 8 or more, 1 uppercase, 1 lowercase, 1 symbol
def password_checker():
    while True:
        password = masked_input("Enter Your New Password: ")

        requirements = [
            len(password) >= 8,
            any(c.islower() for c in password),
            any(c.isupper() for c in password),
            any(c in "!@#$%^&*()_-+=~`|\\{[]}:'\";<>,.?/" for c in password)
        ]
        if all(requirements):
            return password

        else:
            Print(('''[red1]
                     Invalid password!
 Please make sure your password contains at least:
 ---> 8 characters
 ---> 1 lowercase letter
 ---> 1 uppercase letter
 ---> 1 symbol
'''), justify = "center")

# VERIFIES IF THE ENTERED PASSWORD IS CORRECT OR NOT
def authorize_passcode(acc_num):
    query = '''SELECT passcode FROM password
    WHERE acc_number = %s'''
    values = [acc_num]
    cursor.execute(query, values)
    passcode = cursor.fetchone()
    return (passcode[0],)




# IT LOCKS OR UNLOCKS AN ACCOUNT
def account_status(acc_num, status):
    acc = get_ainfo(acc_num)

    # DEACTIVATES THE ACCOUNT AND DISABLES IT FROM BEING USED
    if status == 'locked':
        query = '''UPDATE account SET status = 1
        WHERE acc_number = %s'''
        values = [acc_num]
        cursor.execute(query, values)
        idpass.commit()

        query = '''INSERT INTO log (acc_number, transaction_type, amount)
        VALUES (%s, %s, %s)'''
        values = (acc_num, 'Locked', acc[1])
        cursor.execute(query,values)
        idpass.commit()

    # ALLOWS THE ACCOUNT TO BE USED BY REACTIVATING IT
    elif status == 'unlocked':
        query = '''UPDATE account SET status = 0
        WHERE acc_number = %s'''
        values = [acc_num]
        cursor.execute(query, values)
        idpass.commit()

        query = '''INSERT INTO log (acc_number, transaction_type, amount)
        VALUES (%s, %s, %s)'''
        values = (acc_num, 'UnLocked', acc[1])
        cursor.execute(query, values)
        idpass.commit()

# RESETS THE PASSWORD
def reset_passcode(acc_num, new_pass):
    query = '''UPDATE password SET passcode = %s
    WHERE acc_number = %s'''
    values = [new_pass, acc_num]
    cursor.execute(query, values)
    idpass.commit()

# CHECKS WHETHER THE ACCOUNT STATUS IS UNLOCKED OR LOCKED
def return_status(acc_num):
    query = '''SELECT status FROM account
    WHERE acc_number = %s'''
    values = [acc_num]
    cursor.execute(query, values)
    result = cursor.fetchone()
    if result == (1,):
        return 1
    else:
        return 0
# ────── DEPOSIT, WITHDRAW, DONATE ──────────────────────────────────────────────

def update_balance(acc_num, new_bal, transaction_type, amt):
# it updates the balance

# RECORDS THE WITHDRAWN/DEPOSITED CASH
    query = '''UPDATE account SET balance = %s
    WHERE acc_number = %s'''
    values = (new_bal, acc_num)
    cursor.execute(query, values)
    idpass.commit()

# LOGS THE WITHDRAWN CASH
    if transaction_type == 'withdrawn':
        query = '''INSERT INTO log (acc_number, transaction_type, amount)
        VALUES (%s, %s, %s)'''
        values = (acc_num, 'withdrawn', amt)
        cursor.execute(query, values)
        idpass.commit()

# LOGS THE DEPOSITED CASH
    elif transaction_type == 'deposited':
        query = '''INSERT INTO log (acc_number, transaction_type, amount)
        VALUES (%s, %s, %s)'''
        values = (acc_num, 'deposited', amt)
        cursor.execute(query, values)
        idpass.commit()

# RECORDS IF USER IS DONATING
def donated(acc_num):

    query = '''SELECT id, toa FROM log
    WHERE acc_number = %s
    ORDER BY id DESC
    LIMIT 1'''
    values = (acc_num ,)
    cursor.execute (query, values)
    result = cursor.fetchone()

    query = '''INSERT INTO charity (transaction_id, donated_time, acc_number)
    VALUES (%s, %s, %s)'''
    values = (result[0], result[1], acc_num)
    cursor.execute(query, values)
    idpass.commit()
    print ("Successfully Donated to Charity")
```

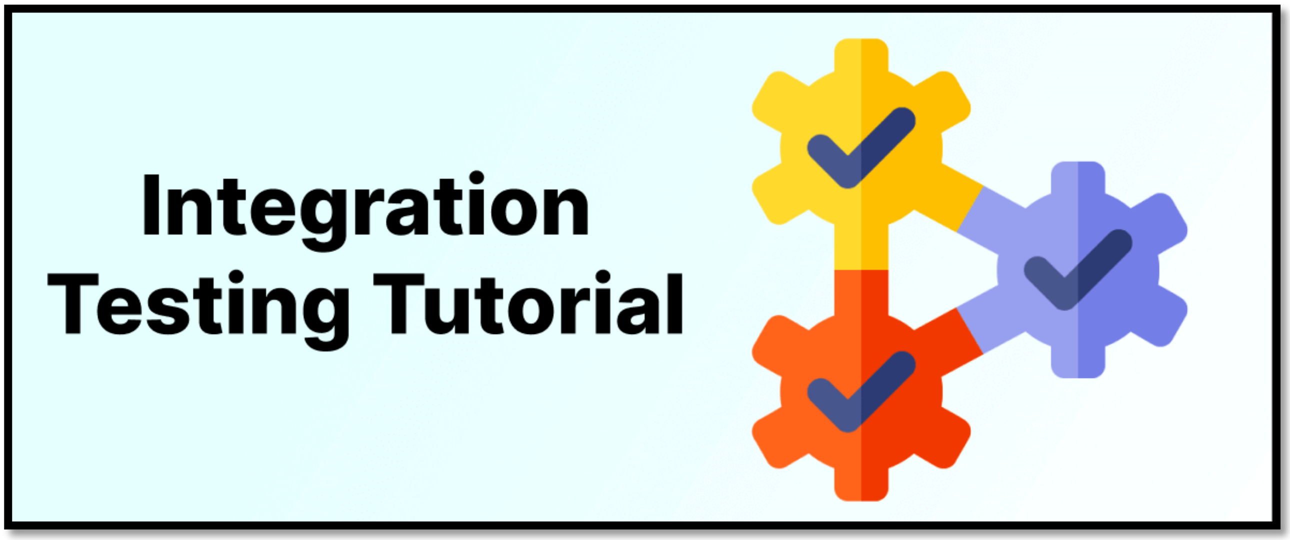
```
# ─────── TRANSFERS ───────────────────────────────────────────────────────────

def update_multiple_balance(s_acc_num, r_acc_num, amt, s_bal, r_bal):

    # RECORDING IN THE SOURCE ACCOUNT
    query = '''UPDATE account SET balance = %s
    WHERE acc_number = %s'''
    values = (s_bal, s_acc_num)
    cursor.execute(query, values)
    idpass.commit()

    # RECORDING IN THE DESTINATION ACCOUNT
    query = '''UPDATE account SET balance = %s
    WHERE acc_number = %s'''
    values = (r_bal, r_acc_num)
    cursor.execute(query, values)
    idpass.commit()

    # FIRST LOG ENTRY
    query = '''INSERT INTO log (acc_number, transaction_type, amount)
    VALUES (%s, %s, %s)'''
    values = (s_acc_num, 'transferred from', amt)
    cursor.execute(query, values)
    idpass.commit()

    # SECOND LOG ENTRY
    query = '''INSERT INTO log (acc_number, transaction_type, amount)
    VALUES (%s, %s, %s)'''
    values = (r_acc_num, 'transferred to', amt)
    cursor.execute(query, values)
    idpass.commit()

# ─────── RETRIEVING DETAILS ──────────────────────────────────────────────────

# GET'S A SINGLE CUSTOMER'S ACCOUNT DETAILS
def get_ainfo(acc_num):
    query = '''SELECT * FROM account
    WHERE acc_number = %s'''
    values = (acc_num,)
    cursor.execute(query, values)
    saccount = cursor.fetchone()
    return saccount

# GET'S A SINGLE CUSTOMER'S PERSONAL INFORMATION
def get_pinfo(acc_num):
    query = '''SELECT * FROM customer_detail
    WHERE acc_number = %s'''
    values = (acc_num,)
    cursor.execute(query, values)
    sper = cursor.fetchone()
    return sper
```

```
# GET'S A SINGLE CUSTOMER'S LOG ENTRIES
def get_linfo(acc_num):
    query = '''SELECT * FROM log
    WHERE acc_number = %s'''
    values = (acc_num,)
    cursor.execute(query, values)
    column_names = [col[0] for col in cursor.description]
    rows = cursor.fetchall()
    return column_names, rows

# GET'S THE ACCOUNT DETAILS AND PERSONAL DETAILS OF ALL THE CUSTOMER'S IN THE
BANK
def get_all_infos(acc_num):
    query = '''SELECT account.acc_number, customer_detail.dob,
    customer_detail.name, customer_detail.gender,
    account.balance, account.status
    FROM account
    INNER JOIN customer_detail
    ON account.acc_number = customer_detail.acc_number
    WHERE account.acc_number = %s'''
    values = (acc_num,)
    cursor.execute(query, values)

    column_names = [col[0] for col in cursor.description]
    rows = cursor.fetchall()
    return column_names, rows

# GET'S THE ACCOUNT DETAILS AND PERSONAL DETAILS OF ALL THE CUSTOMER'S IN THE
BANK
def get_all_info():
    query = '''
    SELECT account.acc_number, customer_detail.dob,
    customer_detail.name, customer_detail.gender,
    account.balance, account.status
    FROM account
    INNER JOIN customer_detail
    WHERE account.acc_number = customer_detail.acc_number
    '''
    cursor.execute(query, )
    column_names = [col[0] for col in cursor.description]
    rows = cursor.fetchall()
    return column_names, rows

# CALCULATES THE TOTAL AMOUNT OF CASH IN THE BANK
def get_total_balance():
    query = 'SELECT * FROM account'
    cursor.execute(query)
    result = cursor.fetchall()
    df = pd.DataFrame(result, columns=['account_id', 'balance', 'status'])
    balance_list = df['balance'].tolist()
    # Convert the 'balance_list' to a list of numbers
    balance_list = [float(value) for value in balance_list]

    # Calculate the sum of all elements in the 'balance_list'
    total_balance = sum(balance_list)
    return total_balance

    # GET'S ALL CUSTOMER'S LOG ENTRIES
def get_all_linfo():
    query = 'SELECT * FROM log'
    cursor.execute(query,)
    column_names = [col[0] for col in cursor.description]
    rows = cursor.fetchall()
    return column_names, rows

# GET'S ALL PAST CUSTOMERS
def pcust_info():
    query = 'SELECT * FROM past_customer'
    cursor.execute(query,)
    column_names = [col[0] for col in cursor.description]
    rows = cursor.fetchall()
    return column_names, rows

# ―――――― DELETING AN ACCOUNT ――――――――――――――――――――――――――――――――――――――――――

def del_acc(acc_num):
    a = get_ainfo(acc_num)
    p = get_pinfo(acc_num)

    # RECORDS ALL DATA IN PAST CUSTOMERS TABLE
    query = '''
    INSERT INTO past_customer(acc_number, dob, name, gender)
    VALUES (%s, %s, %s, %s)
    '''
    values = (a[0], p[1], p[2], p[3])

    # Execute the SQL query and commit the transaction
    cursor.execute(query, values)
    idpass.commit()

    # RECORDS IN LOG
    query = '''
    INSERT INTO log (acc_number, transaction_type, amount)
    VALUES (%s, %s, %s)
    '''
    values = (acc_num, 'withdrawn and deleted', a[1],)

    # Execute the SQL query and commit the transaction
    cursor.execute(query, values)
    idpass.commit()
    # DELETE THE ACCOUNT'S PASSWORD
    query = "DELETE FROM password WHERE acc_number = %s"
    values = (acc_num,)

    # Execute the SQL query and commit the transaction
    cursor.execute(query, values)
    idpass.commit()

    # DELETE THE ACCOUNT FROM BANK
    query = "DELETE FROM account WHERE acc_number = %s"
    values = (acc_num,)

    # Execute the SQL query and commit the transaction
    cursor.execute(query, values)
    idpass.commit()

# ────── INTERFACE TOOLS ──────────────────────────────────────────

def clean_terminal_screen():
    """
    Cleans the terminal screen by performing a system
    clear command. Cls on windows and Clear on UNIX ones.
    """
    os.system('cls' if os.name == 'nt' else 'clear')

f = Figlet(font='Standard')

def DrawText(text, center=True):
    if center:
        lines = [x.center(shutil.get_terminal_size().columns) for x
                 in f.renderText(text).split("\n")]
    else:
        lines = f.renderText(text).split("\n")
    return lines

def box1():
    # Draw the text "Phoenix Bank" and add it to a Panel
    phoenix_bank_text = DrawText("Phoenix Bank")
    phoenix_bank_panel = Text("\n".join(phoenix_bank_text),
style="dark_goldenrod")
    # Create a Table for Main Menu
    table = Table(title="Main Menu", title_style="medium_spring_green",
style="yellow4"
                 , box=box.ROUNDED)
    table.add_column("Option", header_style="orange3", style="royal_blue1")
    table.add_column("Opt_Num", justify="left", header_style="orange3"
                     , style="hot_pink3")
    # Add rows to the table
    options = [("Create new account", "1"),
        ("View account details", "2"),
        ("Deposit", "3"),
        ("Withdraw", "4"),
        ("Transfer", "5"),
        ("Manager's menu", "6"),
        ("Delete account", "7"),
        ("Exit", "8")]
    for option in options:
        table.add_row(*option)
    # Print the title and then the table
    Print(phoenix_bank_panel)
    Print(table, justify="center")

# DISPLAYS THE MAIN DETAILS OF A SINGLE ACCOUNT
def disp_single_acc_info(acc_num):
    account = get_ainfo(acc_num)
    name = get_pinfo(acc_num)
    account_number = str(account[0])
    account_balance = str(account[1])
    table = Table(show_header=False,
                  title="[dark_olive_green2]Your Account Details:")
    table.add_column("[dark_goldenrod]Field", style="bold")
    table.add_column("[dark_goldenrod]Value")
    table.add_row("[sandy_brown]Account Number", account_number)
    table.add_row("[sandy_brown]Account Holder", name[2])
    table.add_row("[sandy_brown]Balance", account_balance)
    Print(table, justify="center")
    print_horizontal_line()

# DISPLAYS THE CURRENT BALANCE WHEN WITHDRAWN, DEPOSITED AND TRANSFERED
def disp_curr_balance(acc_num):
    account = get_ainfo(acc_num)
    Print(Panel("[dark_goldenrod]Your Current Balance --->", account[2])
                  , justify = "center")

def locked():
        Print(('''[red1]
┌──────────────────────────────────────────────────┐
│        Exceeded Maximum Number of Tries          │
│             Your Account is Locked.              │
└──────────────────────────────────────────────────┘'''),justify = "center")
```

```
def box4():

    # Draw the text "Phoenix Bank" and add it to a Panel
    title_text = DrawText("Phoenix Bank")
    title_panel = Text("\n".join(title_text), style="dark_goldenrod")

    # Create a Table for Manager's Menu
    table = Table(title="Manager's Menu", title_style="medium_spring_green"
                  , style="yellow4", box=box.ROUNDED)
    table.add_column("Option", header_style="orange3", style="royal_blue1")
    table.add_column("Opt_Num", justify="left", header_style="orange3"
                     , style="hot_pink3")

    # Add rows to the table
    options = [("View Account Details", "1"),
        ("View All Accounts", "2"),
        ("View All Previous Accounts", "3"),
        ("View Log Entries", "4"),
        ("View Total Cash(Bank)", "5"),
        ("Reset Password", "6"),
        ("Exit", "7")]

    for option in options:
        table.add_row(*option)
    # Print the title and then the table
    Print(title_panel)
    Print(table, justify="center")

# DISPLAY'S THE DATA PROVIDED IN FANCY GRID TABLE FORMAT
def display_tabulated_data(column_names, rows):
    table = tabulate(rows, headers=column_names, tablefmt="fancy_grid")

    # Get terminal width using shutil
    terminal_width, _ = shutil.get_terminal_size()
    Print(table, justify="center")

# ────── EXITING PHOENIX BANK MANAGEMENT SYSTEM ─────────────────────────────────

def break_Off():
    cursor.close()
    clean_terminal_screen()
    sys.exit()

# ────── MAIN PROGRAM ──────────────────────────────────────────────────────────

# CREATING/MAKING SURE THAT ALL THE REQUIRED TABLES ARE CREATED
create_account(), create_customer_detail(), create_password(),
create_log(), create_charity(), create_past_customer()

# MAIN BANKING MENU
while True:
    box1()
    print()
    choice = console.input("[magenta1]Enter your Choice: ")

# ────── CHOICE_1 ──────────────────────────────────────────────────────

    # CREATES A NEW ACCOUNT WITH A RANDOM ACCOUNT NUMBER
    if choice == '1':
        # COLLECTS ALL THE INFORMATION REQUIRED FOR A NEW ACCOUNT
        acc_num = generate_random_number()
        name = pyip.inputStr(
            f"{Fore.LIGHTBLUE_EX}Enter Your Name: {Style.RESET_ALL}")
        dob = input(
            f"{Fore.LIGHTBLUE_EX}Enter Your Date of Birth (dd/mm/year):
{Style.RESET_ALL}")
        gender = input(
            f"{Fore.LIGHTBLUE_EX}Enter Your Gender: {Style.RESET_ALL}")
        passcode = password_checker()
        balance = pyip.inputFloat(
            f"{Fore.LIGHTBLUE_EX}Enter Initial Balance: {Style.RESET_ALL}")

        # REGISTER'S THE ACCOUNT BY ENTERING THE COLLECTED INFORMATION INTO
TABLES
        create_new_account(acc_num, name, dob, gender, balance, passcode)
        Print(Panel("[deep_pink4]Account Created Successfully!")
                     , justify = "center")

    # PRINTS THE MAIN DETAILS OF THE CREATED ACCOUNT TO THE USER
        disp_single_acc_info(acc_num)

# ────── CHOICE_2 ──────────────────────────────────────────────────────

  # ALLOWS THE CUSTOMER's TO VIEW THE BALANCE LEFT IN THEIR ACCOUNTS
    elif choice == '2':

        acc_num = pyip.inputInt(
            f"{Fore.LIGHTBLUE_EX}Enter Account Number: {Style.RESET_ALL}")
        acc = get_ainfo(acc_num)

        # CHECKS IF THE ACCOUNT IS VALID AND UNLOCKED
        if not acc:
            Print(Panel("[red1]Account Not found!"), justify = "center")

        elif return_status(acc_num) == 1:
            locked()
            Print("[red1]Contact Bank Manager to Unlock", justify = "center")

        # AUTHENTICATES THE PASSWORD
        else:
            max_tries = 3
            for _ in range(max_tries):
                passc = masked_input("Enter Your Password: ")

                # PRINTS THE DETAILS IF CORRECT PASSWORD IS ENTERED
                if authorize_passcode(acc_num) == (passc,):
                    disp_single_acc_info(acc_num)
                    break
                else:
                    Print(Panel("[red1]Wrong Password"), justify = "center")

            # ACCOUNT IS LOCKED IF WRONG PASSWORD IS ENTERED 3 TIMES
            else:
                locked()
                account_status(acc_num, 'locked')

# ────── CHOICE_3 ───────────────────────────────────────────────

    # ALLOWS THE CUSTOMER's TO DEPOSIT MONEY
    elif choice == '3':
        acc_num = pyip.inputInt(
            f"{Fore.LIGHTBLUE_EX}Enter Account Number: {Style.RESET_ALL}")
        acc = get_ainfo(acc_num,)

        # CHECKS IF THE ACCOUNT IS VALID
        if acc:
            amt = pyip.inputInt(
                f"{Fore.LIGHTBLUE_EX}Enter Amount to Deposit: {Style.RESET_ALL}")
            Print('''[orange_red1]
┌──────────────────────────────────────────┐
│Would You Like to Donate 1 AED to Charity │
│          1.Yes          2.No             │
└──────────────────────────────────────────┘''', justify="center")
            c = int(input(":"))
            transaction_type = 'deposited'
            new_bal = np.add(amt, acc[1])

            if c == 1:
                amt = np.subtract(amt, 1)
                new_bal = np.subtract(new_bal, 1)
                update_balance(acc_num, new_bal.item(), transaction_type,
amt.item())
                acc = get_ainfo(acc_num)
                donated(acc_num)
                Print(Panel("[deep_pink4]Amount Deposited Successfully!")
                      , justify="center")
                Print("[deep_pink4]Your Current Balance is --->", acc[1]
                      , "[deep_pink4]Dhs", justify="center")

            elif c == 2:
                update_balance(acc_num, new_bal.item(), transaction_type, amt)
                acc = get_ainfo(acc_num)
                Print(Panel("[deep_pink4]Amount Deposited Successfully!")
                  , justify = "center")
               Print("[deep_pink4]Your Current Balance is --->", acc[1]
                  , "[deep_pink4]Dhs", justify = "center")
           else:
               Print(Panel('[red1]Invalid Choice'), justify = "center")

# ────── CHOICE_4 ────────────────────────────────────────────────────────────

   # ALLOWS THE CUSTOMER's TO WITHDRAW THEIR MONEY
    elif choice == '4':

        acc_num = pyip.inputInt(
            f"{Fore.LIGHTBLUE_EX}Enter Account Number: {Style.RESET_ALL}")
        acc = get_ainfo(acc_num)

        # CHECKS IF THE ACCOUNT IS VALID AND UNLOCKED
        if not acc:
            Print(Panel("[red1]Account Not Found!"), justify = "center")
        elif return_status(acc_num) == 1:
            locked()
            Print("[red1]Contact Bank Manager to Unlock", justify = "center")

        # AUTHENTICATES THE PASSWORD
        else:
            for _ in range(3):
                passc = masked_input("Enter Your Password: ")

                # ALLOWS CUSTOMER's TO WITHDRAW IF CORRECT PASSWORD IS ENTERED
                if authorize_passcode(acc_num) == (passc,):

                    amt = pyip.inputInt(
                        f"{Fore.LIGHTBLUE_EX}Enter Amount To Withdraw:
{Style.RESET_ALL}")
                    transaction_type = 'withdrawn'

                    if acc[1] > amt:
                        new_bal = np.subtract(acc[1], amt)
                        update_balance(acc_num, new_bal.item(), transaction_type,
amt)
                        Print(Panel("[deep_pink4]Amount Withdrawn Successfully!")
                              , justify = "center")
                        Print("[deep_pink4]Your Current Balance is --->",
new_bal, "Dhs"
                              , justify = "center")
                        break
                    elif acc[1] == amt:
                        p = '[cornflower_blue]Close Account To Withdraw All Money
From Account'
                        Print(Panel(p)
                              , justify = "center")
                        break
                    else:
                        Print(Panel("[red1]Insufficient Balance!"), justify =
"center")
                        Print("[red1]You Have Only --->", acc[1], "Dhs In Your
Account"
                            , justify = "center")
                        break
                else:
                    Print(Panel("[red1]Wrong Password"), justify = "center")

            # ACCOUNT IS LOCKED IF WRONG PASSWORD IS ENTERED 3 TIMES
            else:
                locked()
                account_status(acc_num, 'locked')

# ──── CHOICE_5 ──────────────────────────────────────────────────────

    # ALLOWS THE CUSTOMER's TO PERFORM BANK TRANSFERS
    elif choice == '5':
        s_acc_num = pyip.inputInt(
            f"{Fore.LIGHTBLUE_EX}Enter Your Account Number: {Style.RESET_ALL}")
        s_acc = get_ainfo(s_acc_num)

        # CHECKS IF THE SOURCE ACCOUNT IS VALID AND UNLOCKED
        if s_acc:
            if return_status(s_acc_num) == 1:
                locked()
                Print("[red1]Contact Bank Manager to Unlock", justify = "center")

            # AUTHENTICATES THE PASSWORD
            else:
                max_tries = 3
                for _ in range(max_tries):
                    passc = masked_input("Enter Your Password: ")

                    # CHECKS IF THE DESTINATION ACCOUNT IS VALID
                    if authorize_passcode(s_acc_num) == (passc,):
                        r_acc_num = pyip.inputInt(
                            f"{Fore.LIGHTBLUE_EX}Enter Destination Account:
{Style.RESET_ALL}")
                        r_acc = get_ainfo(r_acc_num)
                        if r_acc:
                            amt = pyip.inputFloat(
                                f"{Fore.LIGHTBLUE_EX}Enter Amount to Transfer:
{Style.RESET_ALL}")
                            # TRANSFERS THE AMOUNT IF SOURCE ACCOUNT HAS
SUFFICIENT BALANCE
                            if s_acc[1] >= amt:
                                s_bal = np.subtract(s_acc[1] ,amt)
                                r_bal = np.add(r_acc[1], amt)
                                update_multiple_balance(s_acc_num, r_acc_num, amt
                                                  , s_bal.item(),
r_bal.item())
                                Print(Panel("[deep_pink4]Bank Transfer
Successful!")
                                      , justify = "center")
                                s_acc = get_ainfo(s_acc_num)
                                Print("[deep_pink4]Remaining Balance is ---> ",
s_acc[1]
                                      , "[deep_pink4]Dhs", justify ="center")
                                break
                            else:
                                Print(Panel("[red1]Insufficient Balance")
                                      , justify = "center")
                                s_acc = get_ainfo(s_acc_num)
                                Print("[red1]You Have Only ---> ", s_acc[1]
                                      , "[red1]Dhs in Your Account", justify =
"center")
                                break
                        else:
                            Print(Panel("[red1]Destination Account Not Found!")
                                  , justify = "center")
                    else:
                        Print(Panel("[red1]Wrong Password"), justify = "center")
                # ACCOUNT IS LOCKED IF WRONG PASSWORD IS ENTERED 3 TIMES
                else:
                    locked()
                    account_status(s_acc_num, 'locked')
        else:
            Print(Panel("[red1]Source Account Not Found"), justify = "center")

# ────── CHOICE_6 ───────────────────────────────────────────────────────────────

    # ALLOWS THE MANAGER TO MANAGE PHOENIX BANK
    elif choice == '6':
        Print(Panel("[dodger_blue2]Only Bank Manager is Allowed to Access the
Manager's Menu")
              , justify = "center")
        max_tries = 3
        password_verified = False
        for tries_left in range(max_tries, 0, -1):
            if not password_verified:
                password = masked_input("Enter the Bank Manager's Password: ")
                if password == "123":
                    password_verified = True
                    Print(Panel("[slate_blue3]IDENTITY VERIFIED"), justify =
"center")

                    while True:
                        print_horizontal_line()
                        box4()
                        ch = console.input("[magenta1]Enter your choice:")
                        if ch == '1':
                            acc_num = pyip.inputInt(
                                f"{Fore.LIGHTBLUE_EX}Enter Account Number: 
{Style.RESET_ALL}")

                            column_names, rows = get_all_infos(acc_num)
                            display_tabulated_data(column_names, rows)
                            column_names, rows = get_linfo(acc_num)
                            display_tabulated_data(column_names, rows)

                        elif ch == '2':
                            column_names, rows = get_all_info()
                            display_tabulated_data(column_names, rows)

                        elif ch == '3':
                            column_names, rows = pcust_info()
                            display_tabulated_data(column_names, rows)

                        elif ch == '4':
                            column_names, rows = get_all_linfo()
                            display_tabulated_data(column_names, rows)

                        elif ch == '5':
                            tbal = get_total_balance()
                            Print("[deep_pink4]Total Cash Available Now Is -->", 
tbal
                                  , "[deep_pink4]DHS Only", justify = "center")

                        elif ch == '6':
                            acc_num = pyip.inputInt(
                                f"{Fore.LIGHTBLUE_EX}Enter Account Number: 
{Style.RESET_ALL}")
                            acc = get_ainfo(acc_num)
                            if acc:
                                np1 = password_checker()
                                np2 = masked_input("Re-Enter Your New Password: 
")


                                if np1 == np2:
                                    new_pass = np2
                                    status = 'unlocked'
                                    account_status(acc_num, status)
                                    reset_passcode(acc_num, new_pass)
                                    Print("Account Password Changed and is now 
Usable!")

                                else:
                                    Print("Passwords Don't Match.")
                            else:
                                Print("Account Does Not Exist.")

                        # EXIT THE MANAGER's MENU
```

```
                    elif ch == '7':
                        Print("Exiting the Manager's Menu...")
                        print_horizontal_line()
                        input("Press Enter to Exit Manager's Menu...")
                        print()
                        clean_terminal_screen()
                        break

                    else:
                        Print("Invalid Choice")
                        print_horizontal_line()
                        input("Press Enter to Continue...")
                        print()
                        clean_terminal_screen()
                break
            elif password == '@':
                break

            # CALLS THE POLICE IF THE WRONG MANAGER's PASSWORD IS ENTERED 3
TIMES
            else:
                Print(Panel("[red1]Incorrect password"), justify = "center")
                Print('[red1]Tries left:', tries_left - 1, justify =
"center")

        if tries_left == 1:
            Print("[bright_red]Unauthorized access detected. Calling the
police..."
                  , justify = "center")
            cursor.close(), sys.exit

# ────── CHOICE_7 ──────────────────────────────────────────────────────

  elif choice == '7':

        acc_num = pyip.inputInt(
            f"{Fore.LIGHTBLUE_EX}Enter Account Number To Delete:
{Style.RESET_ALL}")
        acc = get_ainfo(acc_num)

        # CHECKS IF THE ACCOUNT IS VALID AND UNLOCKED
        if not acc:
            Print(Panel("[red1]Account Not found!"), justify = "center")

        elif return_status(acc_num) == 1:
            locked()
            Print("[red1]Contact Bank Manager to Unlock", justify = "center")

        # AUTHENTICATES THE PASSWORD
        else:
            max_tries = 3
            for _ in range(max_tries):
                passc = masked_input("Enter Your Password: ")

                if authorize_passcode(acc_num) == (passc,):
                    del_acc(acc_num)

                    Print(Panel("[wheat4]Account Deleted Successfully")
                          , justify = "center")
                    Print(Panel("[dark_olive_green2]sad to see you go")
                          , justify = "center")
                    break
                else:
                    Print(Panel("[red1]Wrong Password"), justify = "center")

            # ACCOUNT IS LOCKED IF WRONG PASSWORD IS ENTERED 3 TIMES
            else:
                locked()
                account_status(acc_num, 'locked')

# ──────  CHOICE_8 ───────────────────────────────────────────────────

    # TO EXIT THE PHOENIX BANK MANAGEMENT SYSTEM
    elif choice == '8':
        input("PRESS ENTER TO EXIT ")
        print_horizontal_line()
        title = pyfiglet.figlet_format('Thank You!', font ='Standard')
        Print(f'[bold blue]{title}',justify ='center')

        for i in track(
            range(3), description = Print("[chartreuse2]Exiting the
application...")):
            time.sleep(1)
        break_Off()
    else:
        Print(Panel("[bold red]Error!"),justify ='center')
        Print(Panel("[bold red]Invalid Choice"), justify = "center")

    print_horizontal_line()
    input("PRESS ENTER TO CONTINUE...")
    clean_terminal_screen()
```

INTEGRATION AND TEST PHASE



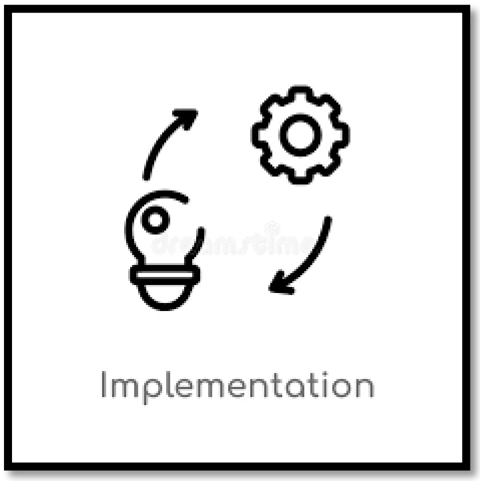
During the integration and test phase, we do several types of testing:

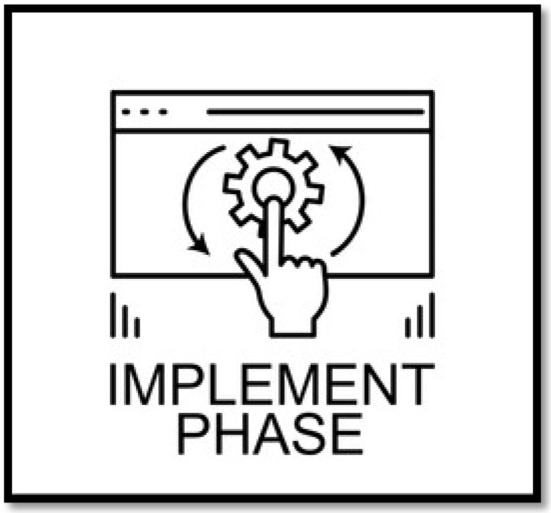
1. Subsystem integration, system, security, and user acceptance testing take place.
2. Users, along with quality assurance teams, make sure that the system meets the functional requirements outlined in the document.
3. The IT security team checks the system's security and gives it a certification before installation.

There are multiple levels of testing:
- Testing at the development facility, often with contractor help and sometimes with end users.
- Testing when the system is in use, with end users working alongside contract personnel.
- Finally, end users perform operational testing, where they use the system for all its functions.

Throughout these tests, we track requirements, perform an Independent Verification & Validation evaluation, and review and accept all documentation before considering the system ready for use.

IMPLEMENTATION PHASE





The implementation phase is the next step in our journey, kicking off once the system successfully passes user testing and gains approval. This phase plays a crucial role in bringing the system to life to support its intended business functions as, It aims to make the system operational for business functions. Key activities include:

1. Performance Evaluation: Comparing system performance against planned objectives.
2. User Communication: Informing users about system changes.
3. Training: Equipping users with skills for effective system use.
4. Hardware Setup: Installing and configuring hardware components.
5. Software Deployment: Installing system software and configuring settings.
6. Workflow Integration: Seamlessly integrating the system into daily operations.
7. Ongoing Monitoring: Ensuring the system operates as per user requirements.

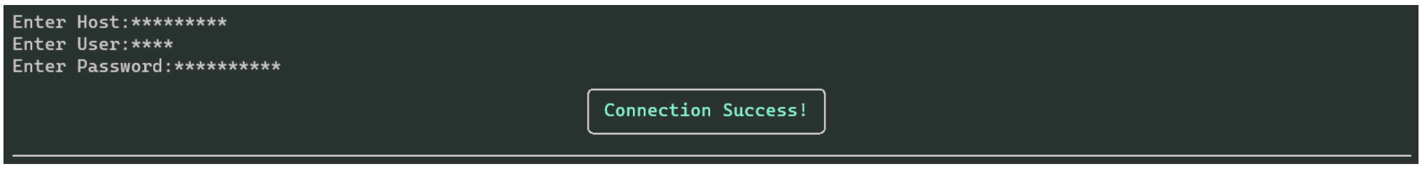
In essence, the implementation phase focuses on transitioning from planning and development to real-world system operation.

Program Output

Login (Establishing Connection)

This part is about SQL,

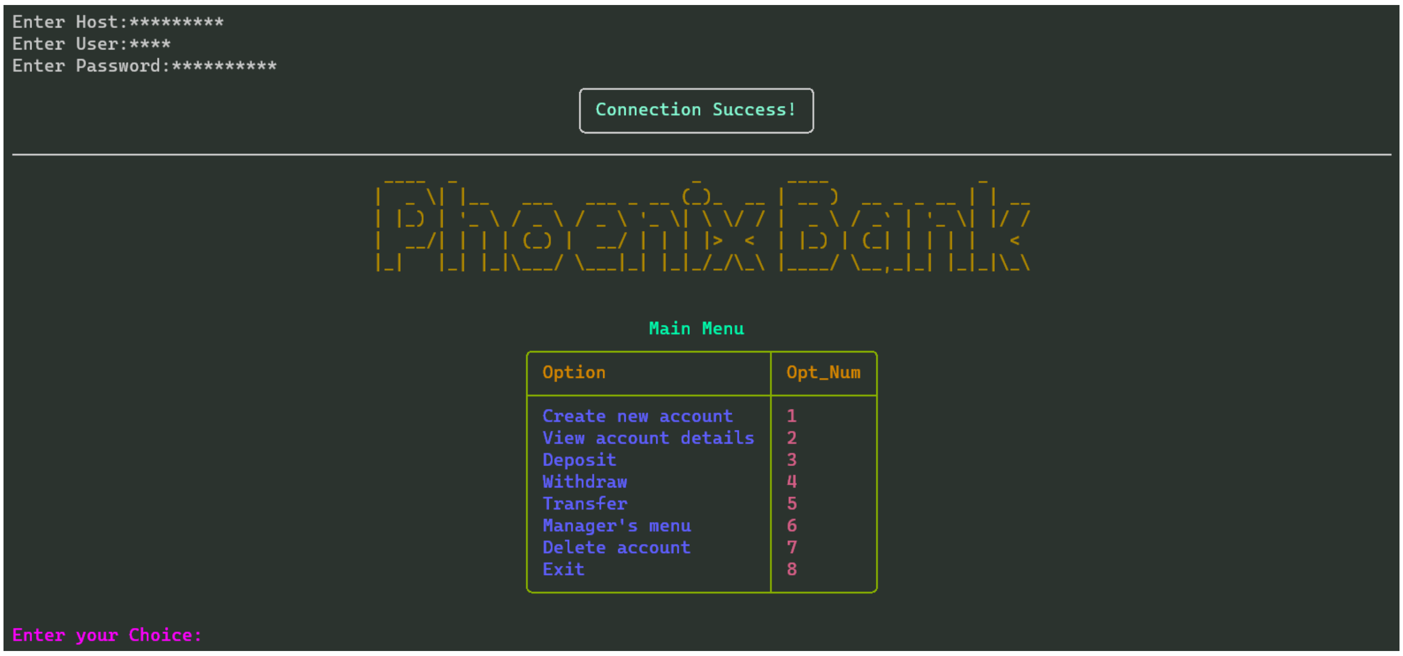
- Kindly insert the accurate information about your SQL



```
Enter Host:*********
Enter User:****
Enter Password:**********

                              Connection Success!
```

Main Menu

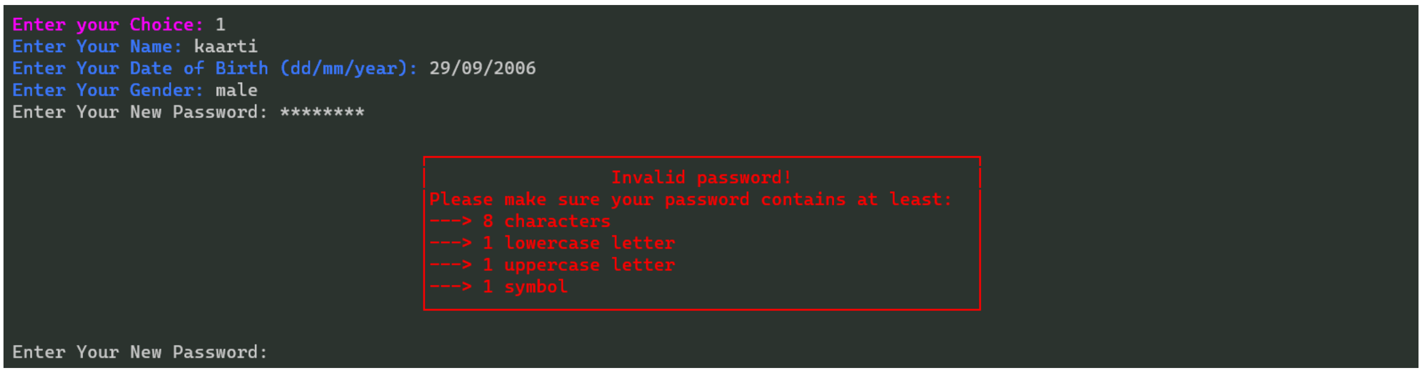
- Main Menu along with the connection part



```
Enter Host:*********
Enter User:****
Enter Password:**********

                              Connection Success!

                  ____  _                      _      ____              _    
                 |  _ \| |__   ___   ___ _ __ (_)_  _| __ )  __ _ _ __ | | __
                 | |_) | '_ \ / _ \ / _ \ '_ \| \ \/ /  _ \ / _` | '_ \| |/ /
                 |  __/| | | | (_) |  __/ | | | |>  <| |_) | (_| | | | |   < 
                 |_|   |_| |_|\___/ \___|_| |_|_/_/\_\____/ \__,_|_| |_|_|\_\

                                  Main Menu

                       Option                 Opt_Num

                       Create new account     1
                       View account details   2
                       Deposit                3
                       Withdraw               4
                       Transfer               5
                       Manager's menu         6
                       Delete account         7
                       Exit                   8

Enter your Choice:
```

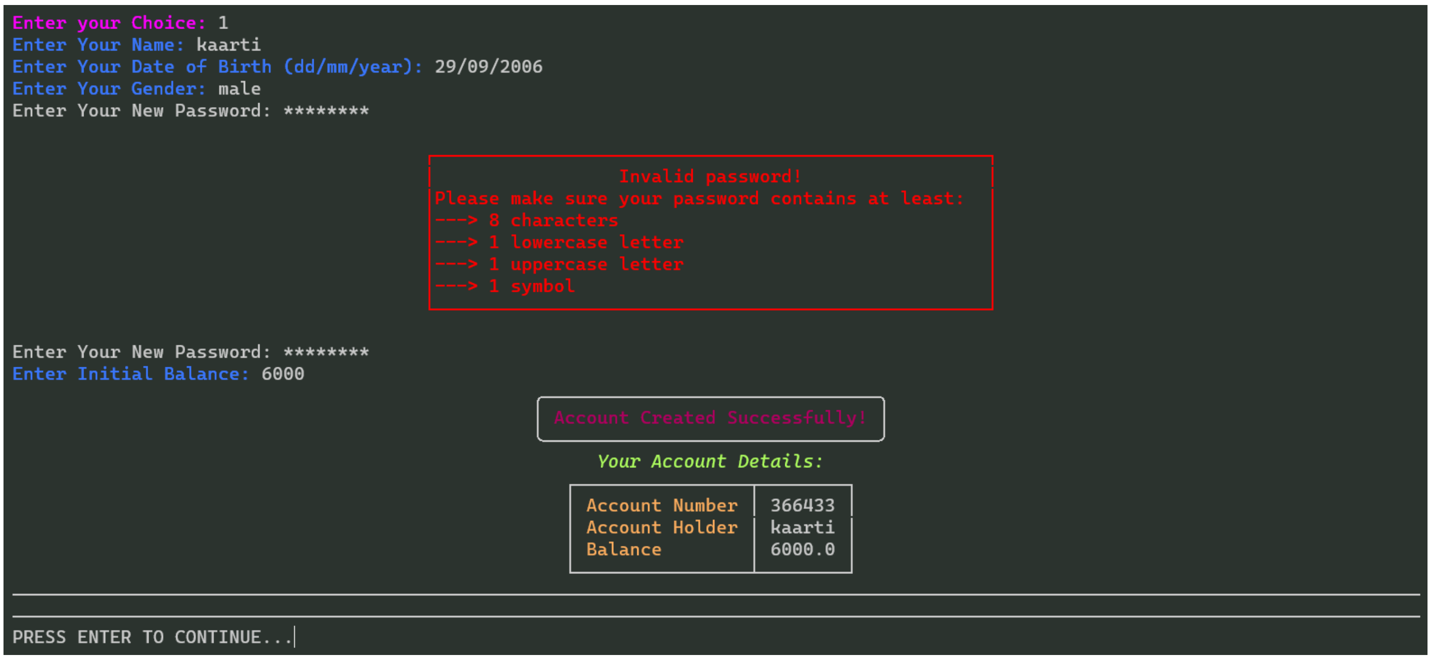
Option_1 (To Create an Account)

- If In Case Password isn't Strong Enough, or It doesn't meet the requirement



```
Enter your Choice: 1
Enter Your Name: kaarti
Enter Your Date of Birth (dd/mm/year): 29/09/2006
Enter Your Gender: male
Enter Your New Password: ********

                                  Invalid password!
                        Please make sure your password contains at least:
                        ---> 8 characters
                        ---> 1 lowercase letter
                        ---> 1 uppercase letter
                        ---> 1 symbol

Enter Your New Password:
```

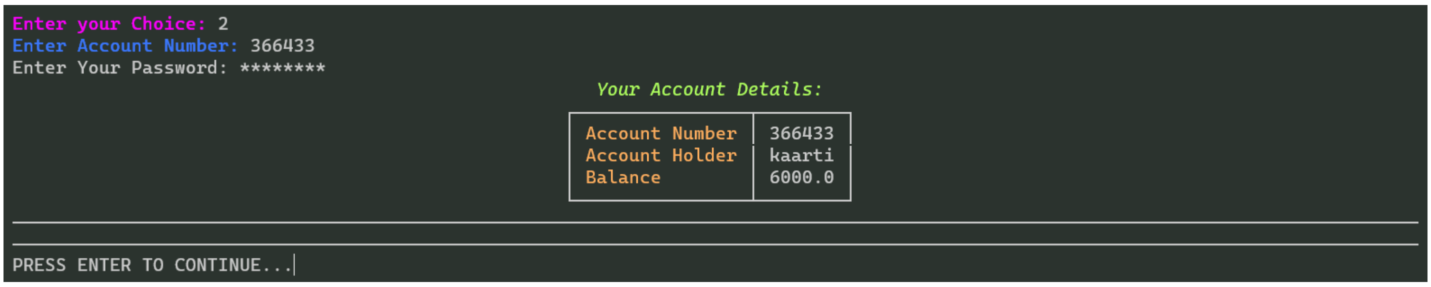
- If Password is Suitable and Strong Enough



```
Enter your Choice: 1
Enter Your Name: kaarti
Enter Your Date of Birth (dd/mm/year): 29/09/2006
Enter Your Gender: male
Enter Your New Password: ********

                                        |            Invalid password!                    |
                                        |Please make sure your password contains at least:|
                                        |---> 8 characters                                |
                                        |---> 1 lowercase letter                          |
                                        |---> 1 uppercase letter                          |
                                        |---> 1 symbol                                    |

Enter Your New Password: ********
Enter Initial Balance: 6000

                                                  Account Created Successfully!

                                                     Your Account Details:

                                                 Account Number | 366433
                                                 Account Holder | kaarti
                                                 Balance        | 6000.0

PRESS ENTER TO CONTINUE...
```

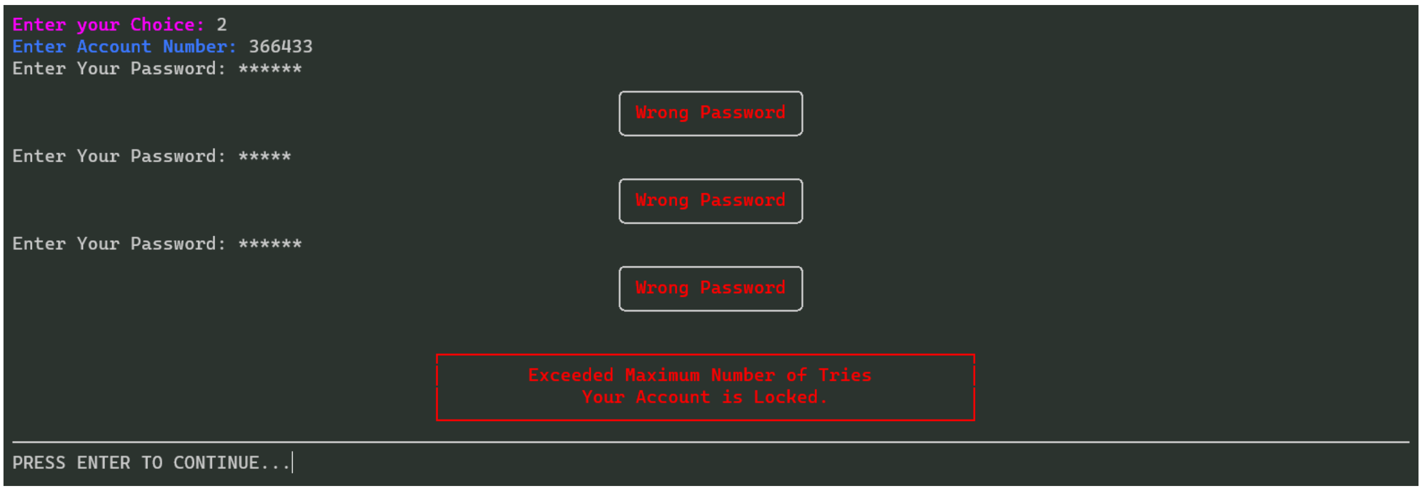
Option_2 (To View the Details of an Account)

- When the correct password is entered



```
Enter your Choice: 2
Enter Account Number: 366433
Enter Your Password: ********
                                                     Your Account Details:

                                                 Account Number | 366433
                                                 Account Holder | kaarti
                                                 Balance        | 6000.0

PRESS ENTER TO CONTINUE...
```

- When the Wrong password is entered

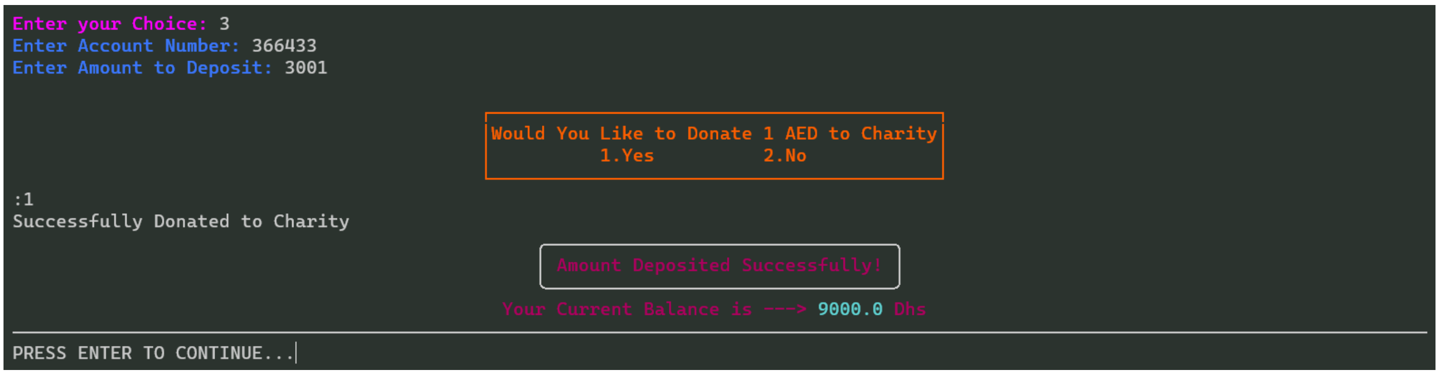


```
Enter your Choice: 2
Enter Account Number: 366433
Enter Your Password: ******
                                                        Wrong Password

Enter Your Password: *****
                                                        Wrong Password

Enter Your Password: ******
                                                        Wrong Password

                                               Exceeded Maximum Number of Tries
                                                   Your Account is Locked.

PRESS ENTER TO CONTINUE...
```

Option_3 (To Deposit into an Account)

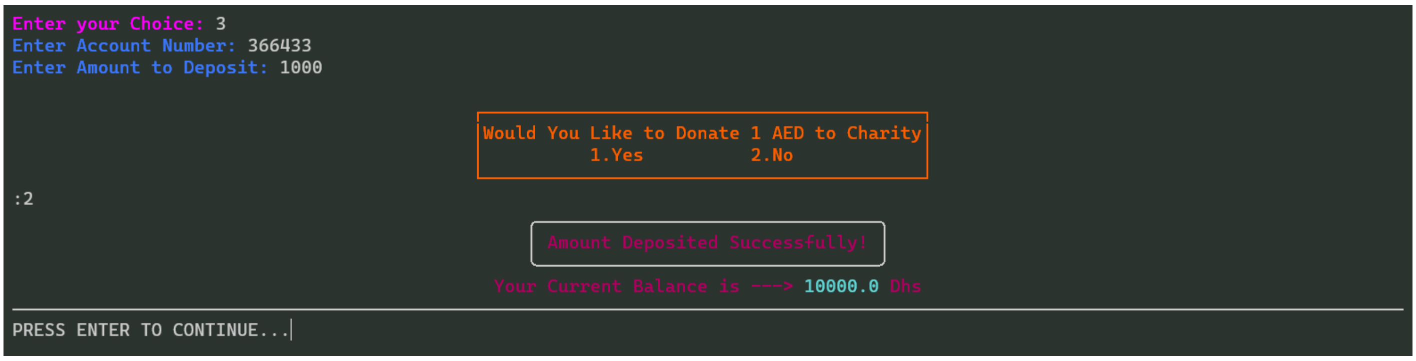
Note – For Depositing into an Account Password is not required (Unlike Withdrawing from an Account)

- If Account Holder Wishes to Donate to Charity



```
Enter your Choice: 3
Enter Account Number: 366433
Enter Amount to Deposit: 3001

                                                Would You Like to Donate 1 AED to Charity
                                                          1.Yes            2.No

:1
Successfully Donated to Charity

                                                     Amount Deposited Successfully!

                                                Your Current Balance is ---> 9000.0 Dhs

PRESS ENTER TO CONTINUE...
```

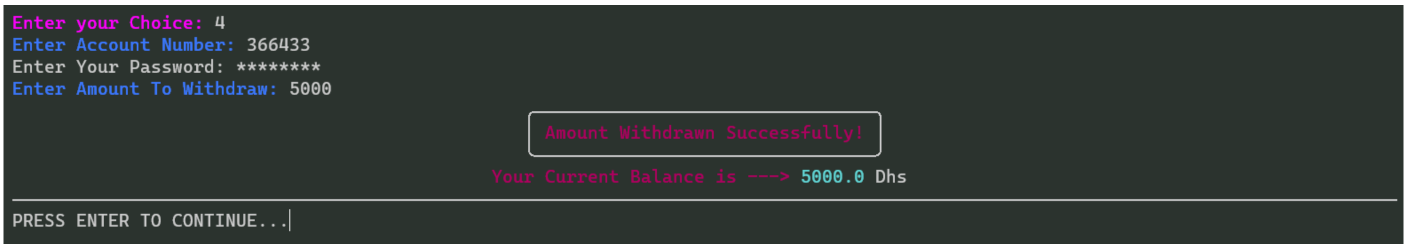
- If Account Holder Doesn't Donate to Charity



```
Enter your Choice: 3
Enter Account Number: 366433
Enter Amount to Deposit: 1000

                                                Would You Like to Donate 1 AED to Charity
                                                          1.Yes            2.No

:2

                                                     Amount Deposited Successfully!

                                                Your Current Balance is ---> 10000.0 Dhs

PRESS ENTER TO CONTINUE...
```

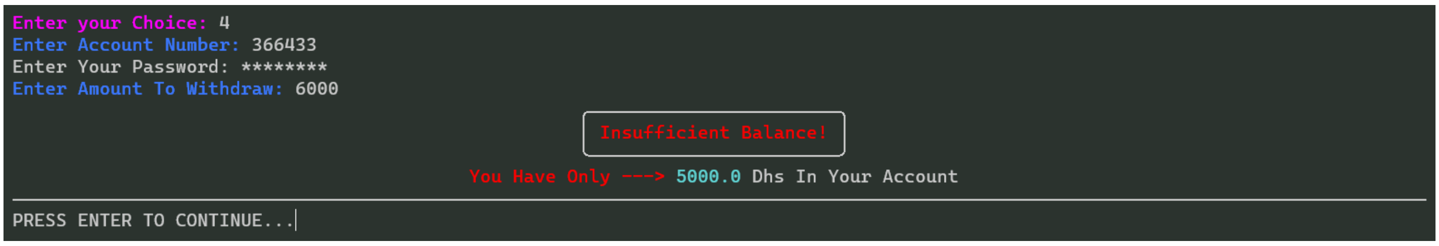
Option_4 (To Withdraw From an Account)

- Withdrawing Amount



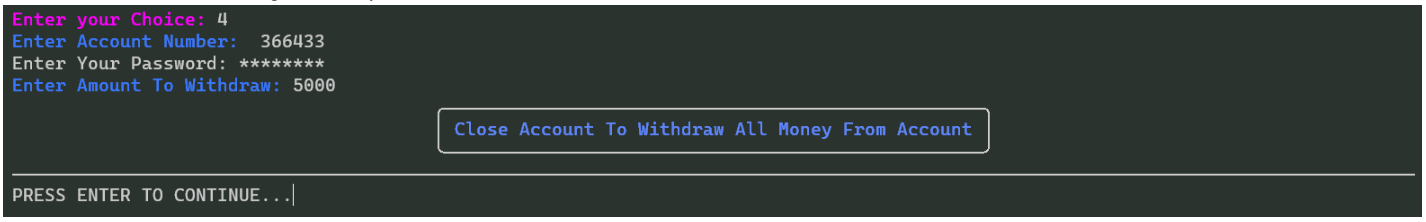
```
Enter your Choice: 4
Enter Account Number: 366433
Enter Your Password: ********
Enter Amount To Withdraw: 5000

                                                     Amount Withdrawn Successfully!

                                                Your Current Balance is ---> 5000.0 Dhs

PRESS ENTER TO CONTINUE...
```

- Withdrawing more Amount than balance in bank



```
Enter your Choice: 4
Enter Account Number: 366433
Enter Your Password: ********
Enter Amount To Withdraw: 6000

                                                         Insufficient Balance!

                                             You Have Only ---> 5000.0 Dhs In Your Account

PRESS ENTER TO CONTINUE...
```

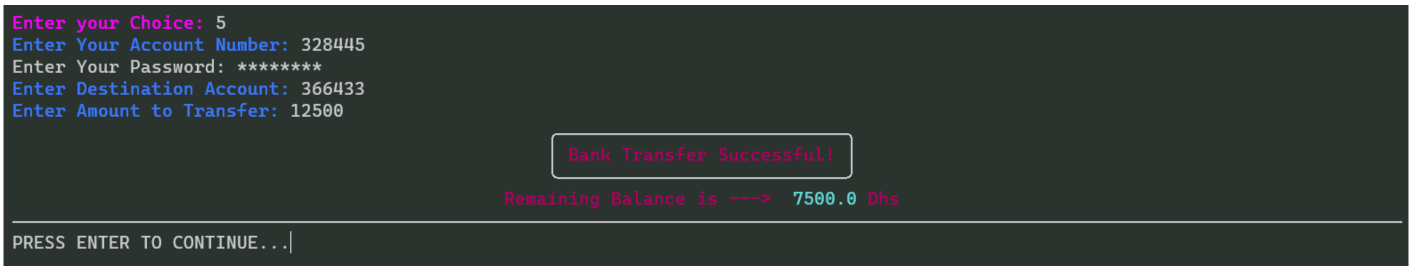
- Withdrawing Exactly Total Amount of balance in Bank



```
Enter your Choice: 4
Enter Account Number:  366433
Enter Your Password: ********
Enter Amount To Withdraw: 5000

                    Close Account To Withdraw All Money From Account

PRESS ENTER TO CONTINUE...
```

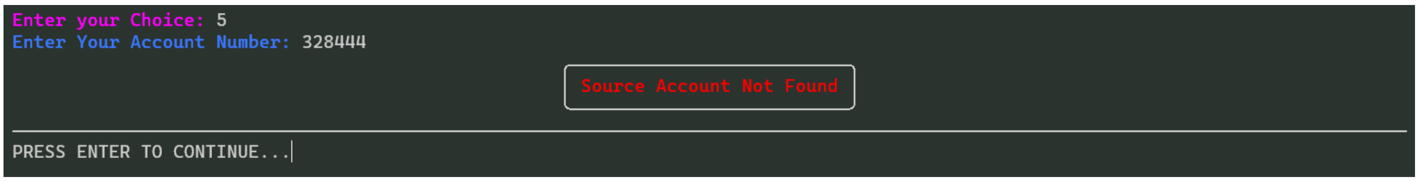
Option_5 (To Perform a Bank Transfer from One Account to Another)

- While both senders and receivers account exists and the sender account has sufficient fund



```
Enter your Choice: 5
Enter Your Account Number: 328445
Enter Your Password: ********
Enter Destination Account: 366433
Enter Amount to Transfer: 12500

                         Bank Transfer Successful!

                  Remaining Balance is --->  7500.0 Dhs

PRESS ENTER TO CONTINUE...
```

- When senders account number is entered wrongly (same occurs when receivers account number is inserted wrongly)

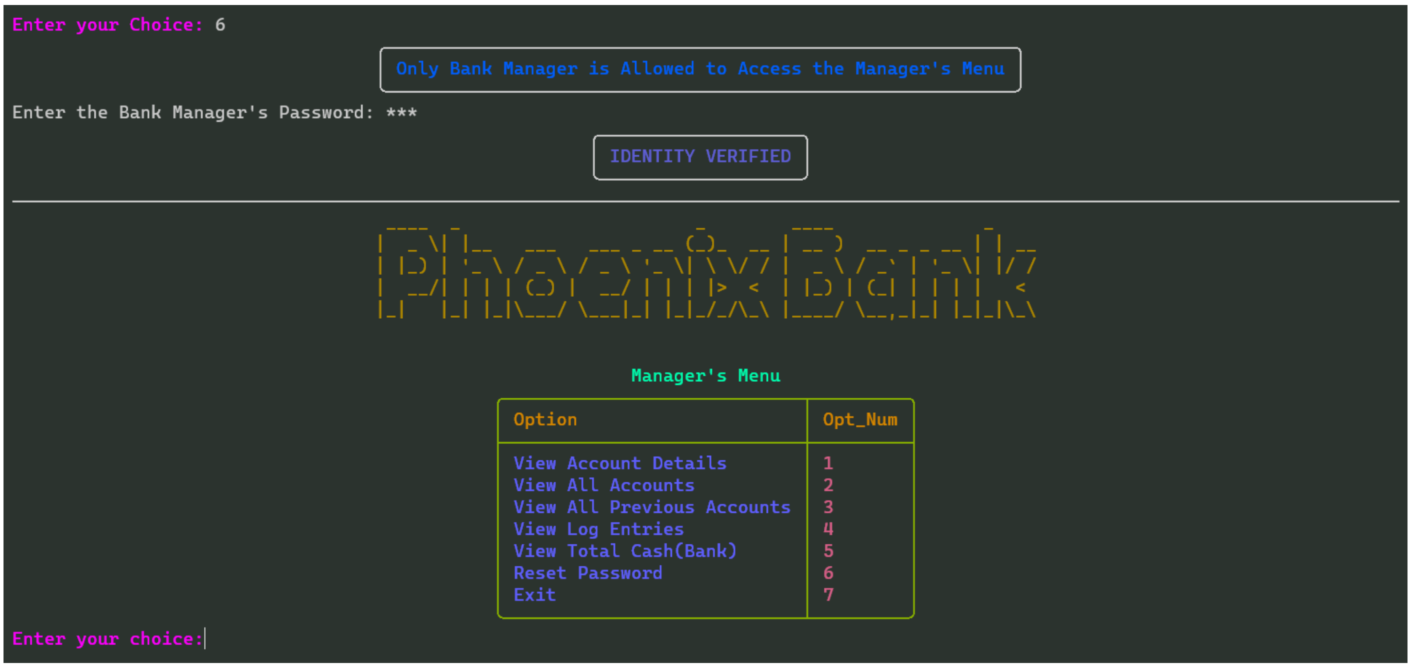


```
Enter your Choice: 5
Enter Your Account Number: 328444

                         Source Account Not Found

PRESS ENTER TO CONTINUE...
```

Option_6 (Manager's Menu)

A Whole Bunch of Exclusive features for Managing the Bank

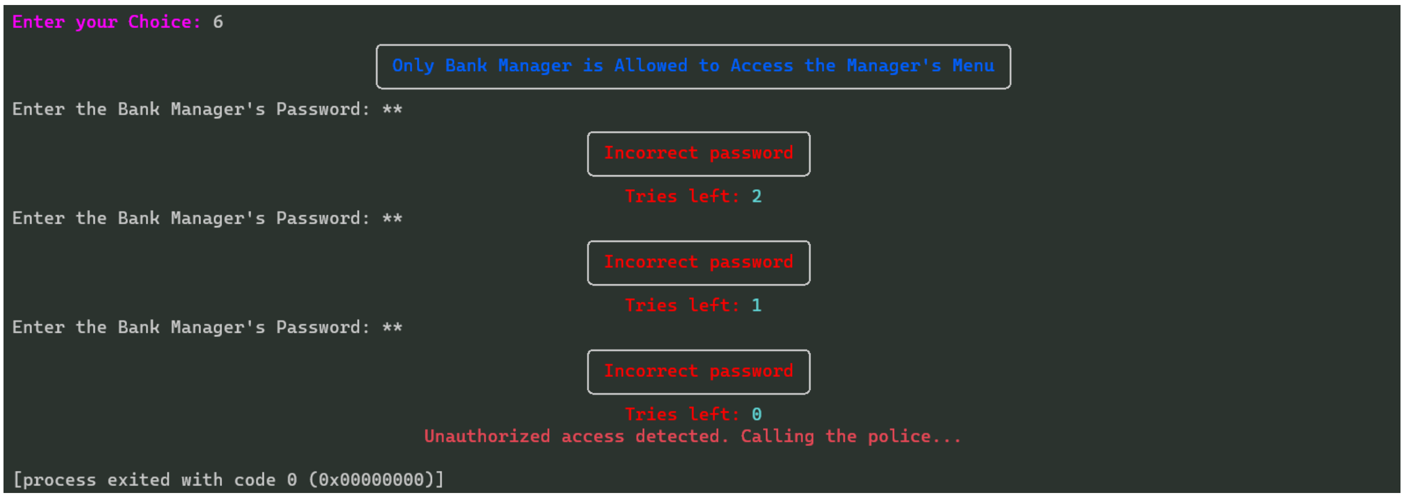
- The managers menu, When correct password is entered



```
Enter your Choice: 6

             Only Bank Manager is Allowed to Access the Manager's Menu

Enter the Bank Manager's Password: ***

                          IDENTITY VERIFIED

                 ____  _                      _      ____              _
                |  _ \| |__   ___   ___ _ __ (_)_  _| __ )  __ _ _ __ | | __
                | |_) | '_ \ / _ \ / _ \ '_ \| \ \/ /  _ \ / _` | '_ \| |/ /
                |  __/| | | | (_) |  __/ | | | |>  <| |_) | (_| | | | |   <
                |_|   |_| |_|\___/ \___|_| |_|_/_/\_\____/ \__,_|_| |_|_|\_\

                            Manager's Menu
                   ┌──────────────────────────┬─────────┐
                   │ Option                   │ Opt_Num │
                   ├──────────────────────────┼─────────┤
                   │ View Account Details     │ 1       │
                   │ View All Accounts        │ 2       │
                   │ View All Previous Accounts │ 3     │
                   │ View Log Entries         │ 4       │
                   │ View Total Cash(Bank)    │ 5       │
                   │ Reset Password           │ 6       │
                   │ Exit                     │ 7       │
                   └──────────────────────────┴─────────┘
Enter your choice:
```

- When wrong password is entered for 3 times

It automatically ends the program and proceeds to warn the user that it is about to contact the authorities

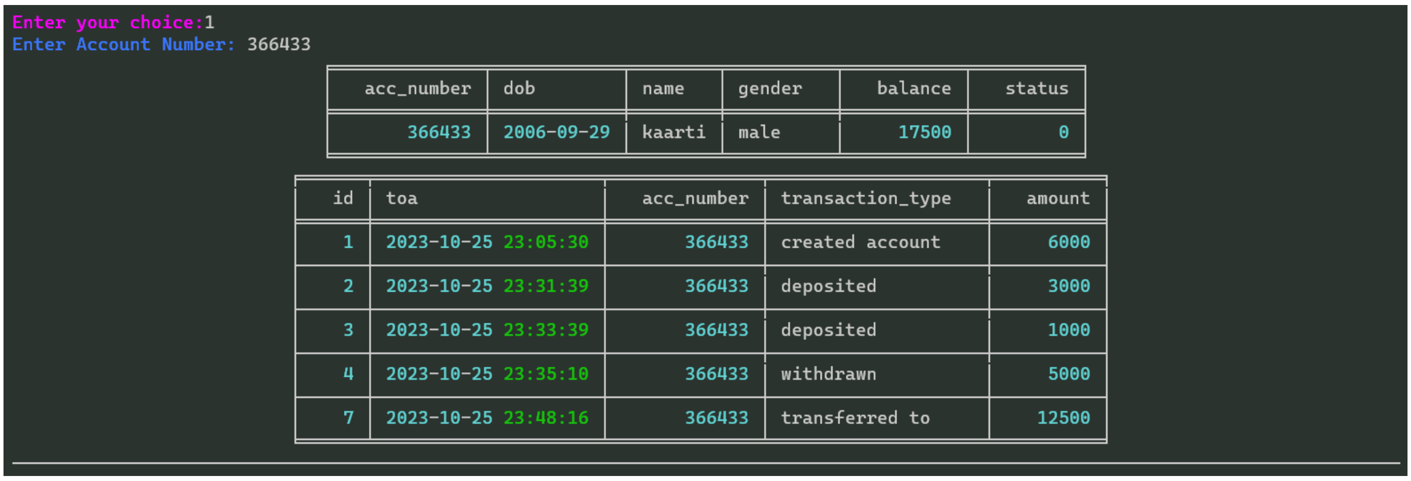


```
Enter your Choice: 6
                           ┌──────────────────────────────────────────────────────────┐
                           │ Only Bank Manager is Allowed to Access the Manager's Menu │
                           └──────────────────────────────────────────────────────────┘
Enter the Bank Manager's Password: **
                                          ┌────────────────────┐
                                          │ Incorrect password │
                                          └────────────────────┘
                                            Tries left: 2
Enter the Bank Manager's Password: **
                                          ┌────────────────────┐
                                          │ Incorrect password │
                                          └────────────────────┘
                                            Tries left: 1
Enter the Bank Manager's Password: **
                                          ┌────────────────────┐
                                          │ Incorrect password │
                                          └────────────────────┘
                                            Tries left: 0
                               Unauthorized access detected. Calling the police...

[process exited with code 0 (0x00000000)]
```

Option_1 (Manager's Menu)

- To view Account Details with Account Specific Log Entries

Here status denotes if the account is locked or not,
(if status is = 0 then it is usable), (if status is = 1 then it is locked)
(The account automatically gets locked when wrong password is entered 3 times)



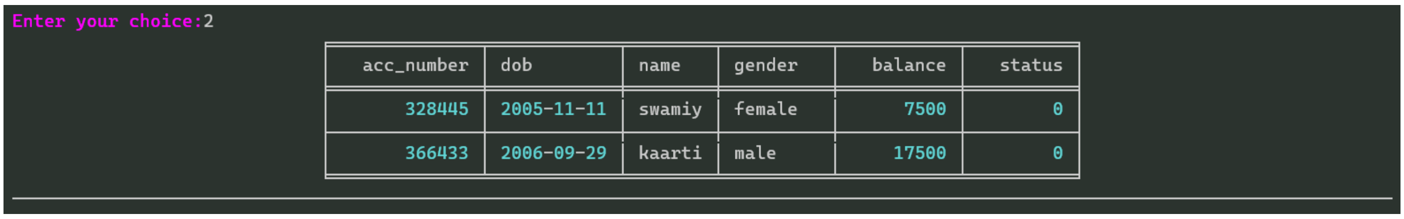
```
Enter your choice:1
Enter Account Number: 366433
```

acc_number	dob	name	gender	balance	status
366433	2006-09-29	kaarti	male	17500	0

id	toa	acc_number	transaction_type	amount
1	2023-10-25 23:05:30	366433	created account	6000
2	2023-10-25 23:31:39	366433	deposited	3000
3	2023-10-25 23:33:39	366433	deposited	1000
4	2023-10-25 23:35:10	366433	withdrawn	5000
7	2023-10-25 23:48:16	366433	transferred to	12500

Option_2 (Manager's Menu)

- To View All The Existing Account Holders In the Bank

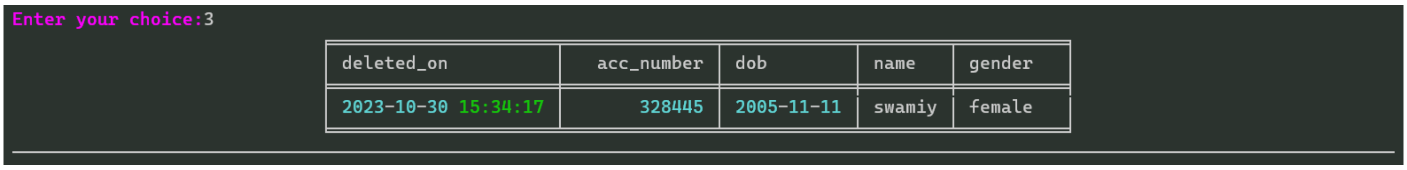


```
Enter your choice:2
```

acc_number	dob	name	gender	balance	status
328445	2005-11-11	swamiy	female	7500	0
366433	2006-09-29	kaarti	male	17500	0

Option_3 (Manager's Menu)

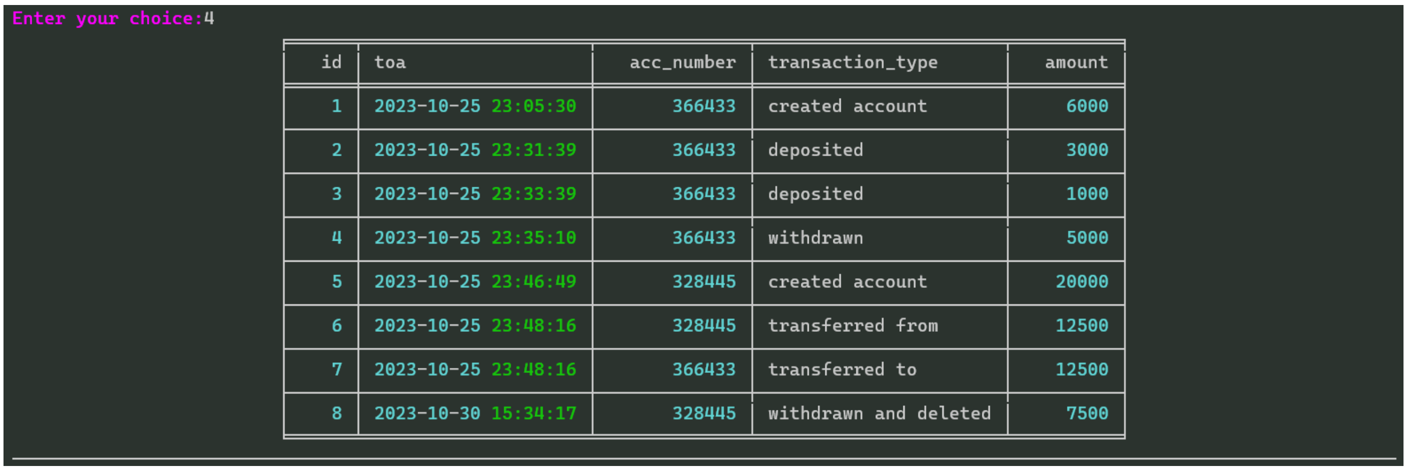
- To view Past Customers/Account Holders



```
Enter your choice:3
                         ┌─────────────────────┬────────────┬────────────┬────────┬──────────┐
                         │ deleted_on          │ acc_number │ dob        │ name   │ gender   │
                         ├─────────────────────┼────────────┼────────────┼────────┼──────────┤
                         │ 2023-10-30 15:34:17 │     328445 │ 2005-11-11 │ swamiy │ female   │
                         └─────────────────────┴────────────┴────────────┴────────┴──────────┘
```

Option_4 (Manager's Menu)

- To view All Recorded Log Entries

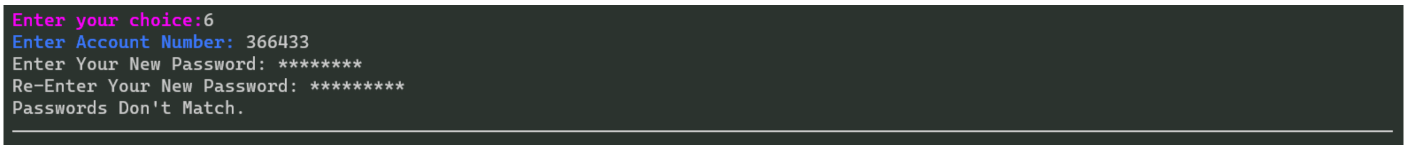


```
Enter your choice:4
                    ┌────┬─────────────────────┬────────────┬────────────────────┬────────┐
                    │ id │ toa                 │ acc_number │ transaction_type   │ amount │
                    ├────┼─────────────────────┼────────────┼────────────────────┼────────┤
                    │  1 │ 2023-10-25 23:05:30 │     366433 │ created account    │   6000 │
                    │  2 │ 2023-10-25 23:31:39 │     366433 │ deposited          │   3000 │
                    │  3 │ 2023-10-25 23:33:39 │     366433 │ deposited          │   1000 │
                    │  4 │ 2023-10-25 23:35:10 │     366433 │ withdrawn          │   5000 │
                    │  5 │ 2023-10-25 23:46:49 │     328445 │ created account    │  20000 │
                    │  6 │ 2023-10-25 23:48:16 │     328445 │ transferred from   │  12500 │
                    │  7 │ 2023-10-25 23:48:16 │     366433 │ transferred to     │  12500 │
                    │  8 │ 2023-10-30 15:34:17 │     328445 │ withdrawn and deleted │  7500 │
                    └────┴─────────────────────┴────────────┴────────────────────┴────────┘
```

Option_5 (Manager's Menu)

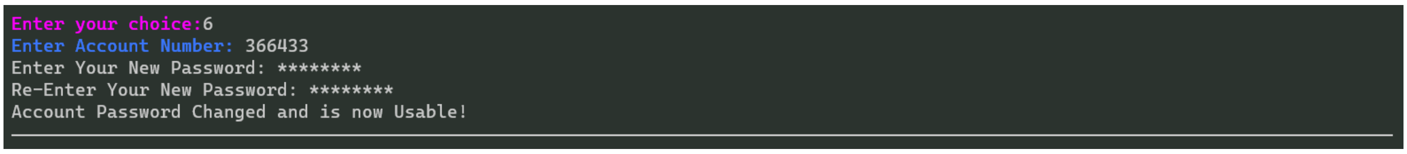
To Reset Password for a locked Account or If Account Holder Forgot Password

- If passwords doesn't match



```
Enter your choice:6
Enter Account Number: 366433
Enter Your New Password: ********
Re-Enter Your New Password: *********
Passwords Don't Match.
```

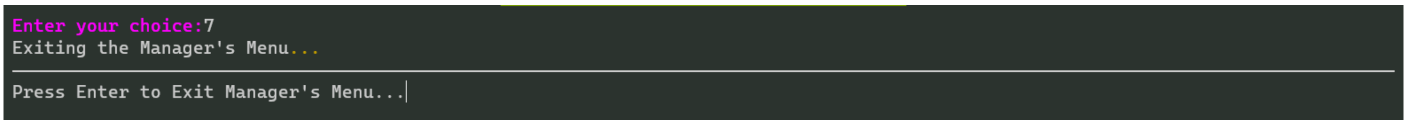
- If passwords match



```
Enter your choice:6
Enter Account Number: 366433
Enter Your New Password: ********
Re-Enter Your New Password: ********
Account Password Changed and is now Usable!
```

Option_6 (Manager's Menu)

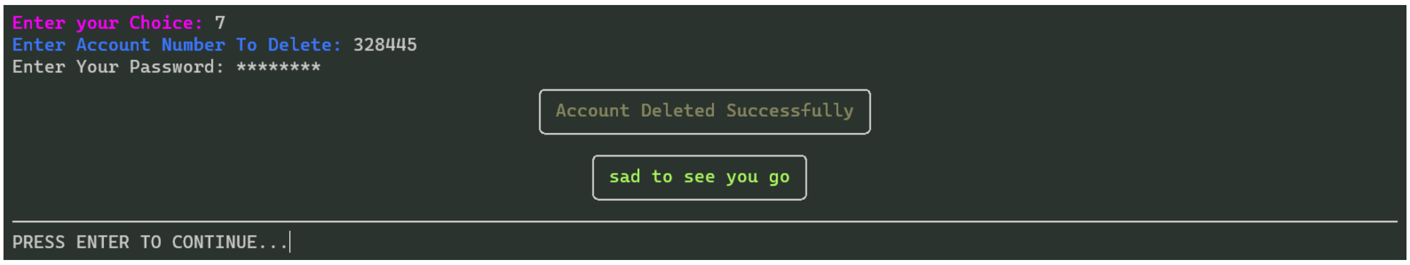
- To Exit Manager's Menu



```
Enter your choice:7
Exiting the Manager's Menu...

Press Enter to Exit Manager's Menu...
```

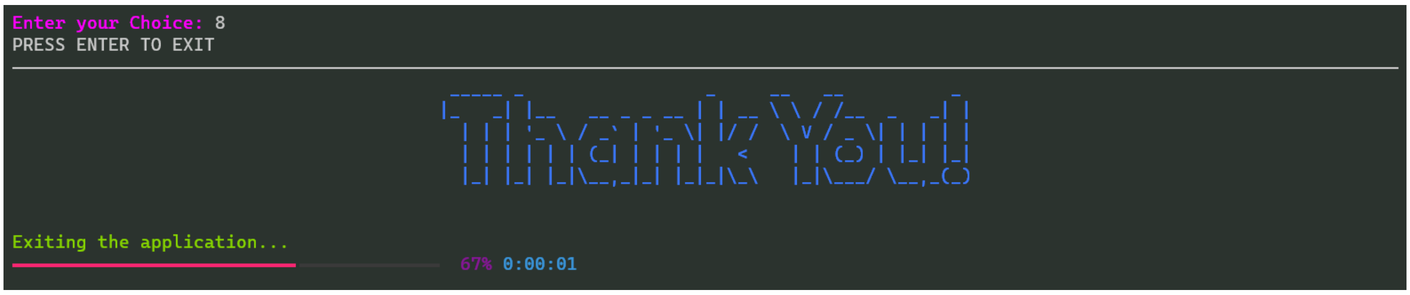
Option_7 (Delete an Account)

- The deleted account's get recorded in the previous account's table



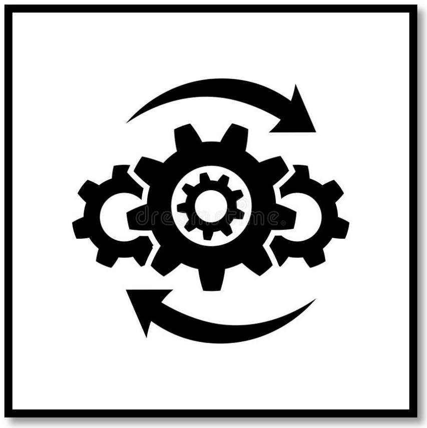
```
Enter your Choice: 7
Enter Account Number To Delete: 328445
Enter Your Password: ********
                                    ╭──────────────────────────────╮
                                    │ Account Deleted Successfully │
                                    ╰──────────────────────────────╯
                                         ╭───────────────────╮
                                         │ sad to see you go │
                                         ╰───────────────────╯
─────────────────────────────────────────────────────────────────────────────────────────────────
PRESS ENTER TO CONTINUE...
```

Option_8 (To Exit Phoenix Bank Management System)



```
Enter your Choice: 8
PRESS ENTER TO EXIT
─────────────────────────────────────────────────────────────────────────────────────────────────
                                      _____ _                 _      __   __          _ 
                                     |_   _| |__   __ _ _ __ | | __  \ \ / /__  _   _| |
                                       | | | '_ \ / _` | '_ \| |/ /   \ V / _ \| | | | |
                                       | | | | | | (_| | | | |   <     | | (_) | |_| |_|
                                       |_| |_| |_|\__,_|_| |_|_|\_\    |_|\___/ \__,_(_)


Exiting the application...
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━  67% 0:00:01
```

OPERATIONS AND MAINTENANCE PHASE



The operational phase is an ongoing phase where the system is continuously monitored to ensure it performs as per user requirements. Modifications are made when necessary to keep the system effective in meeting the organization's evolving needs. This phase continues as long as the system remains adaptable and beneficial.

The objectives of this phase are:

1. System Operation and Enhancement: To maintain, operate, and improve the system, making it a valuable asset for the organization.

2. Security Certification: To verify that the system can securely process sensitive information, ensuring data integrity and confidentiality.

3. Periodic Assessments: To conduct regular evaluations of the system's functionality, ensuring it continues to meet the organization's requirements.

4. Modernization and Retirement: To determine when the system should be modernized, replaced, or retired, aligning it with the organization's changing needs.

In essence, the operational phase ensures the system remains a reliable and effective tool for the organization, adapting to new challenges and opportunities as they arise. When significant changes are required, it may lead to a re-evaluation and potential return to the planning phase.

HARDWARE AND SOFTWARE REQUIREMENTS

Software Requirements:

Operating System:	Window-7 and later versions (32bit, 64 bit) Or any operating system that runs python (full version)
Front End Language:	Python
Back End Language:	SQL
IDE:	Spyder(run file on external console) Or Visual Studio Code
BackEnd:	MySQL

Hardware Requirements:

Processor:	Pentium Dual Core (min) 32bit or 64 bit
Hard Disk:	10GB (min)
Random Access Memory (RAM):	4GB (min)

BIBLIOGRAPHY

Books:

1. Informatics Practices with Python: Textbook for CBSE Class 12 – by Preeti Arora
2. Informatics Practices with Python: Textbook for CBSE Class 12 – by Sumita Arora
3. Informatics Practices Class-XII NCERT Publication

Websites:

1. https://www.wikipedia.org
2. https://www.google.com
3. https://www.youtube.com
4. https://www.geeksforgeeks.org
5. http://python.mykvs.in/index.php

websites used for learning modules, libraries and connection with mysql :

- **https://rich.readthedocs.io/en/stable/**
- **https://mysqlclient.readthedocs.io/**
- **https://pyinputplus.readthedocs.io/en/latest/**
- **https://super-devops.readthedocs.io/en/latest/misc.html**
- **https://pyneng.readthedocs.io/en/latest/book/12_useful_modules/tabulate.html**
- **https://python.readthedocs.io/en/latest/library/shutil.html**

pictures take from

- **https://stock.adobe.com/ae**
- **https://www.pexels.com/**