IU Logo (New)–

Department: Software Engineering Program: **BS**

**SEN-101-L –– Introduction to Python**

|  |  |  |  |
| --- | --- | --- | --- |
| Announced date: \_\_\_\_\_\_\_\_\_\_\_ | Due Date: \_\_\_\_\_\_\_\_\_ | | Total Marks = |
| Teacher Name**: UMAT UL SHAIYA** |  | Marks Obtained = | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Complex Engineering Problem** | | | |
| **Sr. No** | **Course Learning Outcomes in**  **CEP** | **Knowledge Profile** | **Complex Problem Solving** |
| CLO1 |  |  |  |

**Problem:**

Project 1: **Simple Task Scheduler**

Objectives:

To create a desktop application that allows users to schedule tasks and receive reminders at specified times. The application will include a simple graphical user interface (GUI) for ease of use and will use the schedule library for managing task timings.

Features:

1. Task Management:

* Users can add new tasks with a title, description, and scheduled time.
* Users can view a list of all scheduled tasks.
* Users can edit or delete existing tasks.

1. Reminders:

* The application will send reminders to the user at the scheduled times. This can be implemented as desktop notifications.

1. Recurring Tasks:

* Users can set tasks to repeat daily, weekly, or monthly.

Components:

1. GUI:

* Use a GUI library like tinder to create the interface.
* Main windows:
  + Login/Signup window (if user authentication is implemented).
  + Main task management window to add, view, edit, and delete tasks.
* Input fields for task title, description, date, and time.
* Buttons for adding, editing, deleting tasks.
* List view for displaying scheduled tasks.

1. Task Scheduling:

* Use the schedule library to manage task timings.
* Functions to add tasks to the schedule, and to check and trigger reminders at the appropriate times.

1. Notifications:

* Use a library like plyer for cross-platform desktop notifications to remind users of their tasks.

Project 2. **Simple Budget Tracker**:

Develop a budget tracker that allows users to input their income and expenses, categorize them, and visualize their spending with charts. Use libraries like matplotlib or pandas for data handling and visualization.

Project 3 Simple E-commerce Backend:

Build a basic backend for an e-commerce site that handles product listings, user registration, shopping carts, and order processing. No need to database or GUI just develop basic mechanism.

**BANKING MANAGEMENT SYSTEM**

**PROJECT REPORT**

*Submitted in fulfilment of the requirement for the*

*Semester project*



IQRA UNIVERSITY

By

**Michael Ghouri (56116)**

**Anas Awan (60205)**

**Javed Iqbal (68217)**

**Faraz Ahmed (65901)**

**6th SEMESTER**

**DEPARTMENT OF SOFTWARE ENGINEERING**

**CERTIFICATE**



IQRA UNIVERSITY

This is to certify that mini-project report entitled **“Banking Management System”** is confide record of work carried out by **Michael, Anas, Javed & Faraz** during the year 2024 in fulfilment of the Requirement of SEMESTER PROJECT of the Degree of Bachelor of Software Engineering (6th Semester).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**UMAT UL SHAIYA Instructor Signature**

**ACKNOWLEDGEMENT**

We bring out the report of our project work endeavoring gratitude to the Almighty Allah. We have received valuable guidance and help from many people among whom some require special mention.

We take this opportunity to express gratitude to our **UMAT UL SHAIYA** for giving us the opportunity to express our ideas.

Our sincere thanks are also to the lab assistants for the help they rendered to our project. Also our sincere thanks to the **M**anagement of **I**QRA **U**NIVERSITY for providing us with all the facilities for completing the project.

Last but not the least we would like to thank our dear parents & our colleagues for their unending support and valuable suggestions for implementing this project.

## Introduction:

This report outlines the development and functionality of a simple yet functional Bank Management System. The system is implemented in Python and provides basic banking operations including account creation, deposits, withdrawals, balance inquiries, and account details viewing. This system is designed to be easily understandable and extensible for further enhancements.

## Objectives:

The primary objectives of this project were:

* To create a basic bank management system that can manage multiple accounts.
* To implement core banking functions such as deposit, withdrawal, and balance checking.
* To provide account management features like viewing account details and searching for accounts by account number.

## System Design:

### Class: BankAccount

This class represents a bank account and encapsulates the account details and operations that can be performed on an account.

#### Attributes:

* account\_number: The unique identifier for the account.
* name: The name of the account holder.
* balance: The current balance of the account.

#### Methods:

- \_\_init\_\_(self, account\_number, name, initial\_balance=0): Initializes a new bank account with the given details.

- deposit(self, amount): Deposits the specified amount into the account.

- withdraw(self, amount): Withdraws the specified amount from the account if sufficient funds are available.

- display\_balance(self): Displays the current balance of the account.

- display\_account\_details(self): Displays detailed information about the account.

### Class: BankSystem

This class manages multiple bank accounts and provides methods to interact with the accounts.

#### Attributes:

- accounts: A dictionary to store bank accounts, keyed by account number.

#### Methods:

- \_\_init\_\_(self): Initializes the bank system with an empty accounts dictionary.

- create\_account(self, account\_number, name): Creates a new bank account with the given account number and holder's name.

- get\_account(self, account\_number): Retrieves the account object for the given account number.

- display\_all\_accounts(self): Displays details of all accounts in the system.

### Function: main

This function provides a menu-driven interface for the user to interact with the bank system.

#### Menu Options:

1. Create Account

2. Deposit

3. Withdraw

4. Check Balance

5. Look Account Details

6. Search Account by Number

7. Display All Accounts

8. Exit

## Code Implementation:

class BankAccount:

def \_init\_(self, account\_number, name, initial\_balance=0):

self.account\_number = account\_number

self.name = name

self.balance = initial\_balance

def deposit(self, amount):

if amount > 0:

self.balance += amount

print(f"Deposited {amount}. New balance is {self.balance}.")

else:

print("Deposit amount must be positive.")

def withdraw(self, amount):

if amount > 0 and amount <= self.balance:

self.balance -= amount

print(f"Withdrew {amount}. Remaining balance is {self.balance}.")

else:

print("Invalid withdrawal amount or insufficient funds.")

def display\_balance(self):

print(f"Account: {self.name}, Balance: {self.balance}")

def display\_account\_details(self):

print(f"Account Number: {self.account\_number}, Account Holder: {self.name}, Balance: {self.balance}")

class BankSystem:

def \_init\_(self):

self.accounts = {}

def create\_account(self, account\_number, name):

if account\_number not in self.accounts:

self.accounts[account\_number] = BankAccount(account\_number, name)

print(f"Account created for {name}.")

else:

print("Account already exists.")

def get\_account(self, account\_number):

return self.accounts.get(account\_number, None)

def delete\_account(self, account\_number):

if account\_number in self.accounts:

del self.accounts[account\_number]

print(f"Account {account\_number} deleted.")

else:

print("Account not found.")

def display\_all\_accounts(self):

if self.accounts:

for account in self.accounts.values():

account.display\_account\_details()

else:

print("No accounts in the system.")

def main():

bank = BankSystem()

while True:

print("""

========================================

| WELCOME TO THE BANK |

========================================

| 1. Create Account |

| 2. Deposit |

| 3. Withdraw |

| 4. Check Balance |

| 5. Look Account Details |

| 6. Search Account by Number |

| 7. Display All Accounts |

| 8. Delete Account |

| 9. Exit |

========================================

""")

choice = input("Enter your choice: ")

if choice == "1":

account\_number = input("Enter account number: ")

name = input("Enter account holder name: ")

bank.create\_account(account\_number, name)

elif choice == "2":

account\_number = input("Enter account number: ")

account = bank.get\_account(account\_number)

if account:

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

account.deposit(amount)

else:

print("Account not found.")

elif choice == "3":

account\_number = input("Enter account number: ")

account = bank.get\_account(account\_number)

if account:

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

account.withdraw(amount)

else:

print("Account not found."

elif choice == "4":

account\_number = input("Enter account number: ")

account = bank.get\_account(account\_number)

if account:

account.display\_balance()

else:

print("Account not found.")

elif choice == "5":

account\_number = input("Enter account number: ")

account = bank.get\_account(account\_number)

if account:

account.display\_account\_details()

else:

print("Account not found.")

elif choice == "6":

account\_number = input("Enter account number to search: ")

account = bank.get\_account(account\_number)

if account:

account.display\_account\_details()

else:

print("Account not found.")

elif choice == "7":

bank.display\_all\_accounts()

elif choice == "8":

account\_number = input("Enter account number to delete: ")

bank.delete\_account(account\_number)

elif choice == "9":

print("Exiting the system.")

break

else:

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

if \_name\_ == "\_main\_":

main()

import csv

import pandas as pd

import matplotlib.pyplot as plt

# Step 1: Create the CSV file

data = [

["Name", "Reg\_ID", "Working"],

["Javed Iqbal", 68217, '33%'],

["Michael Ghouri", 56116, '33%'],

["Anas Awan", 60205, '33%'],

]

with open('StudentWorking.csv', 'w', newline='') as file:

writer = csv.writer(file)

writer.writerows(data)

print("CSV file 'StudentWorking.csv' created successfully.")

print(" ")

print("\*\*\* Student Data \*\*\*\*")

print(" ")

# Step 2: Read the CSV file into a DataFrame

df = pd.read\_csv('StudentWorking.csv')

# Check the content of the DataFrame

print(df)

# Step 3: Plot the Data

plt.figure(figsize=(10,6))

# Define the background bars with light blue color representing the full 100%

plt.bar(df['Name'], 100, color='lightblue', label='Total Participate')

# Overlay the working hours with orange color

plt.bar(df['Name'], df['Working'], color='orange', label='Participate of work')

# Adding title and labels

plt.title('Percentage of Participate for Each Member')

plt.xlabel('Name')

plt.ylabel('Percentage of participate in project')

plt.legend()

# Show plot

plt.show()

Conclusion:

The Bank Management System developed by our group provides essential banking operations in a simple and user-friendly manner. It supports multiple accounts and offers features to manage and view account details efficiently. This system can be a foundation for more complex banking applications and can be extended with additional functionalities as needed.