PROJECT ON

POST OFFICE ACCOUNT SYSTEM

MADE BY

Name: Saumya Mishra,Anushka

Roll Number: 17,24

Class: 12th

School: PM SHRI Kendriya Vidyalaya No.1 Cantt, Shahjahanpur

Under: Kendriya Vidyalaya Sangathan (KVS)

Subject Teacher: Vaibhav Sir

INDEX

CERTIFICATE

This is to certify that **Saumya Mishra**,**Anushka** student of Class XII, Kendriya Vidyalaya No.1 Cantt, Shahjahanpur, has successfully completed the Computer Science project titled **"Post Office Account System"** under my guidance.

Teacher's Signature: _____

Date: _____

Acknowledgement

I would like to express my sincere gratitude to my Computer Science teacher **Vaibhav Sir** for his constant guidance, encouragement, and valuable suggestions throughout the development of this project. His support and motivation helped me to understand the concepts of Python programming and MySQL database clearly.

I am also thankful to the authorities of **PM SHRI Kendriya Vidyalaya No.1 Cantt, Shahjahanpur** for providing the necessary facilities and resources required for the successful completion of this project.

I would like to extend my appreciation to my parents and friends for their support and encouragement during the course of this work.

Lastly, I thank everyone who directly or indirectly contributed to the completion of this project.

1. Overview

The Post Office Account Management System is a Python-based application designed to manage savings schemes and customer accounts in postal banking operations. The system facilitates three primary deposit schemes: Savings Bank (SB), Recurring Deposit (RD), and Time Deposit (TD), with comprehensive customer relationship management and secure authentication mechanisms.

1.1 Core Objectives

- Enable customer account creation with Aadhaar-based identification
- Process deposits, withdrawals, and interest calculations
- Manage three distinct banking schemes with scheme-specific operations
- Provide secure role-based user authentication
- Facilitate PDF form distribution for banking transactions

---

2. System Architecture

2.1 Technology Stack

The application utilizes **Python 3.x** as the primary programming language with the following key components:

- **mysql.connector**: Database connectivity to MySQL 5.7+ (port 3307)
- **Decimal Module**: High-precision financial calculations (replaces float)
- **Hashlib**: SHA-256 password hashing for security
- **Webbrowser & OS Modules**: PDF form management and file handling

2.2 Database Structure

The system requires four primary tables in the "post_office" MySQL database:

Users Table: Stores authentication credentials

- Columns: username, password_hash, role, status (ACTIVE/INACTIVE)
- Purpose: Role-based access control for staff

Customers Table: Maintains customer master records

- Columns: customer_id (CIF), aadhaar, name, address, mobile
- Key Feature: Aadhaar as unique identifier to prevent duplicate registrations

Accounts Table: Stores account information

- Columns: acc_no, customer_id, name, address, mobile, acc_type, balance, status
- Account Types: SB (Savings Bank), RD (Recurring Deposit), TD (Time Deposit)
- Status Values: Active, Closed

RD Details Table: RD scheme-specific tracking

- Columns: acc_no, monthly_amount, months_completed
- Purpose: Manages installment tracking for recurring deposits

2.3 Account Number Scheme

The system uses 12-digit account numbers with scheme-specific prefixes:

Scheme	Prefix	Example	Description
SB	010	010123456789	Savings Bank Account
RD	020	020987654321	Recurring Deposit Account
TD	030	030555666777	Time Deposit Account

Customer Identification Numbers (CIF) are 9-digit unique identifiers generated at account creation[1].

---

3. User Authentication System

3.1 Login Mechanism

The login function implements a three-attempt security protocol:

1. **Username & Password Validation**: Users input credentials (maximum 3 attempts)
2. **Password Hashing**: User input is hashed using SHA-256 and compared against stored hashes
3. **Database Query**: SELECT query retrieves user role and account status
4. **Status Verification**: Only ACTIVE accounts are permitted login
5. **Session Management**: Returns user object containing username and role

Security Implementation:
Input Password → SHA-256 Hash → Database Comparison → Account Status Check

3.2 Role-Based Access Control

The system supports role-based authorization through the "role" field in the users table. The application structure allows for customized menu access based on user roles (e.g., admin, teller, officer).

---

4. Customer Account Management

4.1 Customer Identification System (CIF)

The CIF (Customer Identification) model enables unified account management:

Key Features:

- One customer can hold multiple accounts (SB, RD, TD simultaneously)
- Aadhaar (12-digit) serves as the primary unique identifier
- Duplicate Aadhaar detection prevents duplicate customer registration
- Existing customer can be referenced using 9-digit CIF

Customer Registration Flow:
Aadhaar Check → Generate Unique 9-digit CIF → Insert into customers table

4.2 Account Creation Process

4.2.1 SB Account (Savings Bank)

- **Minimum Opening Balance**: ₹500 (enforced)
- **Interest Rate**: 4% per annum (compounded, effective 1 Oct–31 Dec 2025)
- **Operations Allowed**: Unlimited deposits, withdrawals (subject to minimum balance)
- **Restrictions**: Only one active SB account per customer

4.2.2 RD Account (Recurring Deposit)

- **Monthly Installment**: Fixed amount determined at account opening
- **Tenure**: 60 months (5 years) mandatory
- **Interest Rate**: 6.7% per annum (compounded monthly)

- **Maturity Condition**: Full maturity after 60 installments
- **Premature Closure**: Allowed after 36 months with reduced benefits

4.2.3 TD Account (Time Deposit)

- **Deposit Type**: One-time lump sum (no subsequent deposits)
- **Tenure**: Fixed at 12 months (1 year)
- **Interest Rate**: 6.9% per annum (compounded)
- **Maturity Rule**: After 1 year, amount automatically transfers to linked SB account and TD closes

---

5. Core Banking Operations

5.1 Deposit Operation

Applicable To: SB accounts only (RD uses dedicated monthly deposit function)

Process:

1. Account number input and validation
2. Account status check (must be Active)
3. Account type verification (SB only)
4. Amount input with error handling
5. Balance update: $Balance_{new} = Balance_{old} + Amount$
6. Database commit

Validation Checks:

- Account existence verification
- Active status enforcement
- SB type enforcement

5.2 Withdrawal Operation

Applicable To: SB accounts only

Mathematical Constraints:

- Withdrawal Amount ≤ Available Balance
- Post-Withdrawal Balance ≥ ₹500 (minimum balance rule)

Precision Handling: Uses Python Decimal module to avoid floating-point rounding errors in financial calculations[2].

Process:

1. Account lookup with balance fetch
2. Status and type validation
3. Amount validation (>0 and ≤ available)
4. Balance adequacy check
5. Minimum balance verification: $Balance - Withdrawal \geq 500$
6. Update: $Balance_{new} = Balance - Withdrawal$

5.3 Balance Enquiry

Function: Retrieves current account status including:

- Account holder name
- Current balance
- Account status (Active/Closed)

Query Type: Simple SELECT without modifications

5.4 Account Closure

Restrictions:

- Only accounts in Active status can be closed
- Closure sets account status to 'Closed'
- Balance is not transferred (must be withdrawn manually)

---

6. Interest Calculation System

6.1 Simple Interest (SB & TD Accounts)

The system implements compound interest using the formula:

$$A = P\left(1 + \frac{r}{100}\right)^{t}$$

Where:

- A = Final Amount

- P = Principal Amount
- r = Annual Interest Rate (%)
- t = Time in years (months ÷ 12)

Supported Periods:

- SB: User-specified months (4% annual rate)
- TD: Fixed 12 months (6.9% annual rate)

Implementation:

- Input: Balance, account type, tenure
- Calculation: Compound growth via exponentiation
- Update: New balance = A, Interest earned = A − P
- Database commit with new balance

6.2 Recurring Deposit Interest (RD Accounts)

RD uses month-by-month compound interest calculation reflecting the accumulating principal[3]:

Algorithm:
For each month (1 to months_completed):
balance += monthly_amount
interest = (balance × rate) / 1200
balance += interest

Key Parameters:

- Monthly Amount: Fixed at account creation
- Rate: 6.7% annual (0.557% monthly equivalent)
- Compounding Frequency: Monthly (divisor = 1200)

Interest Eligibility:

- Full Interest: After 60 months
- Partial Interest (Reduced Benefit): Between 36–59 months
- No Interest: Before 36 months (premature closure not allowed)

---

7. Recurring Deposit Scheme Management

7.1 Monthly RD Deposit Function

Purpose: Accumulate monthly installments toward the 60-month tenure

Validation Logic:

1. Fetch RD account with associated monthly amount and months completed
2. Prevent deposits after 60 installments (account is matured)
3. Validate remaining installments: $Remaining = 60 - months\_completed$
4. Enforce installment limit: $New\_Installments \leq Remaining$

Update Operations:

- Balance Update: $Balance = Balance + (monthly\_amount \times installments)$
- Month Tracking: $months\_completed = months\_completed + installments$

7.2 RD Schedule Generation

Purpose: Display month-by-month progression of deposits and interest

Output Table Columns:

Month	Deposit	Interest	Balance
1	5000	23.33	5023.33
2	5000	46.99	10070.32

Rate Note: Schedule uses fixed 5.8% for demonstration (differs from actual 6.7% maturity calculation)

7.3 RD Maturity Transfer (RD to SB)

Pre-Conditions:

- RD account must have completed minimum 36 months
- Target SB account must exist and be Active
- RD account must be Active

Maturity Calculation:

- Recomputes full compound interest over all completed months
- Formula: Month-by-month accumulation as per 6.7% annual rate

Post-Transfer Actions:

1. Credit SB account with maturity amount
2. Set RD balance to 0

3. Change RD status to 'Closed'
4. Database commit

Output Information:

- Total installments contributed
- Total deposits (without interest)
- Maturity amount (with interest)

---

8. Time Deposit Scheme Management

8.1 TD Maturity Transfer (TD to SB)

Mandatory Completion: After exactly 12 months tenure

Calculation:

- Principal: Original TD deposit amount
- Formula: $Maturity\_Amount = Principal \times (1 + 6.9/100)^1$
- Interest Earned: $Interest = Maturity\_Amount - Principal$

Example Calculation:
Principal: ₹50,000
Rate: 6.9%
Time: 12 months (1 year)
Maturity Amount = 50,000 × $(1.069)^1$ = ₹53,450
Interest Earned = ₹3,450

Transfer Process:

1. Validate TD account (Active status, exists)
2. Validate SB account (Active status, exists)
3. Calculate maturity with 6.9% compound interest
4. Credit SB: $SB\_Balance = SB\_Balance + Maturity\_Amount$
5. Close TD: Status = 'Closed', Balance = 0

---

9. Account Search Functionality

9.1 Search Methods

The system provides four search mechanisms:

9.1.1 Search by Name

- **Query**: LIKE operator for partial matching
- **Example**: "Rah" returns "Raj Kumar", "Radhika Sharma"
- **Output**: All matching accounts with details

9.1.2 Search by Account Number

- **Query**: Exact match on acc_no field
- **Output**: Single account if found

9.1.3 Search by Mobile Number

- **Validation**: 10-digit numeric check
- **Query**: Exact match on mobile field
- **Output**: All accounts linked to mobile

9.1.4 Search by Aadhaar Number (NEW)

- **Validation**: 12-digit numeric check
- **Two-Step Process**:
    a. Query customers table by Aadhaar to find CIF
    b. Query accounts table by CIF to retrieve all linked accounts
- **Output**: Customer details + all associated accounts

CIF Advantage: Single Aadhaar can reveal all accounts (SB + RD + TD) held by customer

---

10. Post Office Forms Management

10.1 Forms Dictionary

The system maintains local PDF paths for standard banking forms:

Form Name	File Path	Applicable Schemes
SB-3 (Savings Account Opening Form)	forms/sb3.pdf	SB
RD Account Opening Form	forms/sb3.pdf	RD
TD Account Opening Form	forms/sb3.pdf	TD

Deposit Slip (SB/RD/TD)	forms/sb103.pdf	All
Withdrawal Form (SB/RD/TD)	forms/sb7.pdf	SB
Account Closure Form (SB)	forms/sb7a.pdf	SB, RD, TD

10.2 Form Delivery Mechanism

File Path Handling:

1. Relative path conversion to absolute path using `os.path.abspath()`
2. File existence validation
3. PDF opened in default browser: `webbrowser.open("file://" + absolute_path)`

Error Handling:

- If file not found: Displays expected path and user instruction
- Creates user-friendly error messages
- Guides users to place PDFs in 'forms/' folder

---

11. Validation Framework

11.1 Aadhaar Validation

- **Criteria**: 12 numeric digits
- **Method**: `aadhaar.isdigit() and len(aadhaar) == 12`
- **Purpose**: Prevents invalid identities and duplicates

11.2 Mobile Number Validation

- **Criteria**: 10 numeric digits
- **Method**: `mobile.isdigit() and len(mobile) == 10`
- **Purpose**: Ensures valid contact information

11.3 Customer ID Validation

- **Criteria**: 9 numeric digits
- **Method**: `customer_id.isdigit() and len(customer_id) == 9`
- **Purpose**: Identifies existing customers

11.4 Amount Validation

- **Criteria**:
  - SB Deposit: ≥ ₹500 minimum balance
  - SB Withdrawal: Maintains ₹500 minimum
  - RD: Positive amount only
  - TD: Positive amount only
- **Implementation**: Try-except blocks with Decimal conversion

---

12. Menu Navigation System

12.1 Main Menu Structure

The application uses nested while-loops to manage hierarchical menus:

Main Menu (10 options)
├── Schemes Menu
│ ├── SB Menu (deposit, withdraw, interest, closure, info)
│ ├── RD Menu (monthly deposit, interest, schedule, maturity, info)
│ └── TD Menu (interest, maturity transfer, info)
└── Other Operations (search, forms, logout)

12.2 Menu Implementation Pattern

Each menu function:

1. Displays options with numbered choices
2. Accepts user input via `input()`
3. Routes to corresponding function via if-elif chain
4. Handles invalid input with error messages
5. Supports back navigation (option 0)
6. Continues until user selects exit

---

13. Data Precision and Financial Calculations

13.1 Decimal Module Usage

To prevent floating-point arithmetic errors in financial calculations, the system uses Python's Decimal module:

Example Scenario:

Float (INCORRECT for finance):

balance = 1000.05
withdrawal = 500.02
remaining = 1000.05 - 500.02 # May yield 500.0299999...

Decimal (CORRECT):

balance = Decimal("1000.05")
withdrawal = Decimal("500.02")
remaining = balance - withdrawal # Yields exactly Decimal("500.03")

13.2 Financial Formula Precision

- **Compound Interest**: Uses exponentiation operator `**` for accurate growth
- **Monthly Interest**: Divides annual rate by 1200 (12 months × 100%)
- **Rounding**: Uses `round(value, 2)` for display purposes only
- **Database Storage**: Maintains full precision in MySQL

---

14. Security Considerations

14.1 Password Security

- **Hashing Algorithm**: SHA-256 one-way hashing
- **Storage**: Password hash stored in database, not plaintext
- **Authentication**: User input hashed and compared against stored hash

14.2 SQL Injection Prevention

- **Parameterized Queries**: All queries use `%s` placeholders
- **Parameter Binding**: Values passed separately to `execute()` method
- **Example**: `cur.execute("SELECT * FROM customers WHERE aadhaar=%s", (aadhaar,))`

14.3 Data Validation

- **Input Sanitization**: All user inputs validated before database operations
- **Type Checking**: Numeric fields verified for digit-only content
- **Length Constraints**: Aadhaar (12), Mobile (10), CIF (9) enforced

---

15. Error Handling and Exception Management

15.1 Database Connection Error

```
try:
con = mysql.connector.connect(...)
except:
print("ERROR: Database not connected")
exit()
```

Purpose: Ensures system exits gracefully if database unavailable

15.2 Transaction Rollback

```
try:
cur.execute(...)
con.commit()
except Exception as e:
con.rollback()
print("Error:", e)
```

Purpose: Maintains data integrity by reverting failed transactions

15.3 Keyboard Interrupt Handling

```
try:
# Main program loop
except KeyboardInterrupt:
print("Exiting program...")
finally:
cur.close()
con.close()
```

Purpose: Ensures clean shutdown and resource cleanup

---

16. Program Execution Flow

16.1 Startup Sequence

1. Import libraries and initialize constants
2. Attempt database connection (exit on failure)
3. Call `login()` function (max 3 attempts)
4. Validate user role and status
5. Call `main_menu()` if authentication successful

16.2 Runtime Loop

1. Display main menu (10 options)
2. Accept user choice
3. Route to appropriate function
4. Execute operation (create account, deposit, etc.)
5. Return to main menu
6. Repeat until logout or exit selected

16.3 Shutdown Sequence

1. User selects option 0 (Exit)
2. Close database cursor
3. Close database connection
4. Exit program

---

17. Scheme Information Display

17.1 SB (Savings Bank) Scheme

- **Account Prefix**: 010xxxxxxxxx (12 digits)
- **Interest Rate**: 4% per annum (compounded)
- **Minimum Balance**: ₹500 (enforced at all times)
- **Operations**: Unlimited deposits/withdrawals subject to minimum balance
- **Tenure**: No fixed tenure (continuous account)
- **Use Case**: General savings and daily transactions

17.2 RD (Recurring Deposit) Scheme

- **Account Prefix**: 020xxxxxxxxx (12 digits)
- **Interest Rate**: 6.7% per annum (compounded monthly)
- **Tenure**: 60 months (5 years) mandatory
- **Monthly Installment**: Fixed at account opening
- **Premature Closure**: Allowed after 36 months (with interest penalty)
- **Use Case**: Disciplined savings with higher returns

17.3 TD (Time Deposit) Scheme

- **Account Prefix**: 030xxxxxxxxx (12 digits)
- **Interest Rate**: 6.9% per annum (compounded)
- **Tenure**: 12 months (1 year) fixed
- **Deposit Type**: One-time lump sum only
- **Maturity Transfer**: Automatic to linked SB account
- **Use Case**: Short-term high-return investment

---

18. Source Code (My sql)

```
DROP DATABASE IF EXISTS post_office;

CREATE DATABASE post_office;

USE post_office;


-- ==========================

-- 1) USERS TABLE (LOGIN)

-- ==========================

CREATE TABLE users (

    username VARCHAR(50) PRIMARY KEY,

    password_hash VARCHAR(64) NOT NULL,

    role ENUM('POSTMASTER','CLERK') NOT NULL DEFAULT 'CLERK',

    status ENUM('ACTIVE','INACTIVE') NOT NULL DEFAULT 'ACTIVE'

);


-- Default login: username = postmaster, password = 1234

-- SHA256("1234") = 03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4

INSERT INTO users(username, password_hash, role, status)

VALUES
('postmaster','03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4','POSTMAS
TER','ACTIVE');


-- ==========================

-- 2) CUSTOMERS TABLE (CIF)

-- ==========================

CREATE TABLE customers (

    customer_id VARCHAR(9) PRIMARY KEY,   -- ✅CIF 9 digits

    aadhaar VARCHAR(12) UNIQUE NOT NULL,   -- ✅1 Aadhaar = 1 Customer

    name VARCHAR(100) NOT NULL,

    address VARCHAR(255) NOT NULL,
```

```
    mobile VARCHAR(10) NOT NULL
);

CREATE INDEX idx_customers_mobile ON customers(mobile);

-- ==========================
-- 3) ACCOUNTS TABLE
-- ==========================
CREATE TABLE accounts (
    acc_no VARCHAR(12) PRIMARY KEY,       -- ✅ 12 digits
    name VARCHAR(100) NOT NULL,
    address VARCHAR(255) NOT NULL,
    mobile VARCHAR(10) NOT NULL,
    acc_type ENUM('SB','RD','TD') NOT NULL,
    balance DECIMAL(12,2) NOT NULL DEFAULT 0.00,
    customer_id VARCHAR(9) NOT NULL,
    status ENUM('Active','Closed') NOT NULL DEFAULT 'Active',
    created_on TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    CONSTRAINT fk_customer
        FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
        ON DELETE CASCADE
);

CREATE INDEX idx_accounts_mobile ON accounts(mobile);
CREATE INDEX idx_accounts_type ON accounts(acc_type);
CREATE INDEX idx_accounts_status ON accounts(status);
CREATE INDEX idx_customer_id ON accounts(customer_id);

-- ✅ RULE: One SB account per customer_id only
-- (RD/TD multiple allowed)
CREATE UNIQUE INDEX unique_one_sb_per_customer
```

```
ON accounts(customer_id, acc_type);


-- BUT above will also restrict RD and TD (wrong) ✗
-- ✓ Fix with trigger-based unique SB-only enforcement ✓

-- Step 1: Create helper column
ALTER TABLE accounts ADD COLUMN sb_unique_key VARCHAR(20) DEFAULT NULL;


DELIMITER $$


-- Insert trigger
CREATE TRIGGER trg_sb_unique_insert
BEFORE INSERT ON accounts
FOR EACH ROW
BEGIN
    IF NEW.acc_type = 'SB' THEN
        SET NEW.sb_unique_key = NEW.customer_id;
    ELSE
        SET NEW.sb_unique_key = NULL;
    END IF;
END$$


-- Update trigger
CREATE TRIGGER trg_sb_unique_update
BEFORE UPDATE ON accounts
FOR EACH ROW
BEGIN
    IF NEW.acc_type = 'SB' THEN
        SET NEW.sb_unique_key = NEW.customer_id;
    ELSE
        SET NEW.sb_unique_key = NULL;
    END IF;
```

```
END$$

DELIMITER ;

-- ✅ Drop wrong unique constraint
DROP INDEX unique_one_sb_per_customer ON accounts;

-- ✅ Create SB-only unique constraint
CREATE UNIQUE INDEX unique_only_sb ON accounts(sb_unique_key);

-- ============================
-- 4) RD DETAILS TABLE
-- ============================
CREATE TABLE rd_details (
    acc_no VARCHAR(12) PRIMARY KEY,
    monthly_amount DECIMAL(10,2) NOT NULL,
    months_completed INT NOT NULL DEFAULT 0,

    CONSTRAINT fk_rd_acc
        FOREIGN KEY (acc_no) REFERENCES accounts(acc_no)
        ON DELETE CASCADE
);

-- ============================
-- 5) TRANSACTIONS TABLE
-- ============================
CREATE TABLE transactions (
    txn_id INT AUTO_INCREMENT PRIMARY KEY,
    acc_no VARCHAR(12) NOT NULL,
    txn_type VARCHAR(30) NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
    txn_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    CONSTRAINT fk_txn_acc
        FOREIGN KEY (acc_no) REFERENCES accounts(acc_no)
        ON DELETE CASCADE
);

CREATE INDEX idx_txn_acc ON transactions(acc_no);
CREATE INDEX idx_txn_type ON transactions(txn_type);
CREATE INDEX idx_txn_date ON transactions(txn_date);

SELECT "✅ post_office database created successfully (Python compatible)" AS STATUS;
```

19. Source Code (python(minimized))

```
import mysql.connector, hashlib, decimal

from decimal import Decimal


# ================ GLOBAL SETUP ================

con = mysql.connector.connect(host="localhost",
database="post_office", port=3307)

cur = con.cursor()


# ================ 1. SECURITY ================

def hash_password(password):

    return hashlib.sha256(password.encode()).hexdigest()


def login():

    for attempt in range(3):

        user = input("Username: ")

        pwd = hash_password(input("Password: "))

        cur.execute("SELECT role FROM users WHERE username=%s AND
password_hash=%s", (user, pwd))

        if cur.fetchone(): return True

    return False


# ================ 2. CUSTOMER SYSTEM ================

def get_customer():
```

```
    cid = input("Customer ID (9 digits) or Enter for NEW: ")
    if cid: return cid  # Existing customer

    # New customer with Aadhaar check
    aadhaar = input("Aadhaar (12 digits): ")
    if len(aadhaar) == 12 and aadhaar.isdigit():
        cid = f"CIF{random.randint(1000000,9999999)}"
        cur.execute("INSERT INTO customers VALUES
(%s,%s,%s,%s,%s)",
                    (cid, aadhaar, name, address, mobile))
        return cid
    return None

# ================ 3. ACCOUNT CREATION ================
def create_account():
    cid = get_customer()
    type = input("SB/RD/TD: ").upper()

    # Generate 12-digit account number
    acc_no =
f"{type_prefix[type]}{random.randint(100000000,999999999)}"

    if type == "SB":
        balance = max(500, float(input("Min ₹500: ")))  # Enforce
min balance
    elif type == "RD":
```

```
        monthly = float(input("Monthly amount: "))

        balance = monthly

    else:  # TD

        balance = float(input("Lump sum: "))


    cur.execute("INSERT INTO accounts VALUES
(%s,%s,%s,%s,%s,%s,%s,'Active')",

                (acc_no, name, address, mobile, type, balance,
cid))

    con.commit()


# ================ 4. SB OPERATIONS ================

def deposit(acc_no):

    cur.execute("SELECT balance,acc_type,status FROM accounts
WHERE acc_no=%s", (acc_no,))

    balance, type, status = cur.fetchone()


    if type == "SB" and status == "Active":

        amt = Decimal(input("Amount: "))

        cur.execute("UPDATE accounts SET balance=balance+%s WHERE
acc_no=%s", (amt, acc_no))

        con.commit()

        print("✅ Deposit successful")


def withdraw(acc_no):

    balance = Decimal(cur.execute("SELECT balance FROM accounts
WHERE acc_no=%s", (acc_no,)).fetchone()[0])
```

```
    amt = Decimal(input("Amount: "))


    if balance - amt >= 500:  # Min balance rule

        cur.execute("UPDATE accounts SET balance=balance-%s WHERE
acc_no=%s", (amt, acc_no))

        con.commit()

        print("✅ Withdrawal successful")

    else:

        print("❌ Min ₹500 balance required")


# ================ 5. INTEREST CALCULATION ================

def compound_interest(acc_no, months):

    cur.execute("SELECT balance,acc_type FROM accounts WHERE
acc_no=%s", (acc_no,))

    balance, type = cur.fetchone()


    rates = {"SB": 4.0, "RD": 6.7, "TD": 6.9}

    rate = rates[type]


    # A = P(1 + r/100)^t

    maturity = balance * (1 + rate/100)**(months/12)

    cur.execute("UPDATE accounts SET balance=%s WHERE acc_no=%s",
(maturity, acc_no))

    print(f"Interest: ₹{maturity-balance:.2f}")


# ================ 6. RD SPECIFIC ================
```

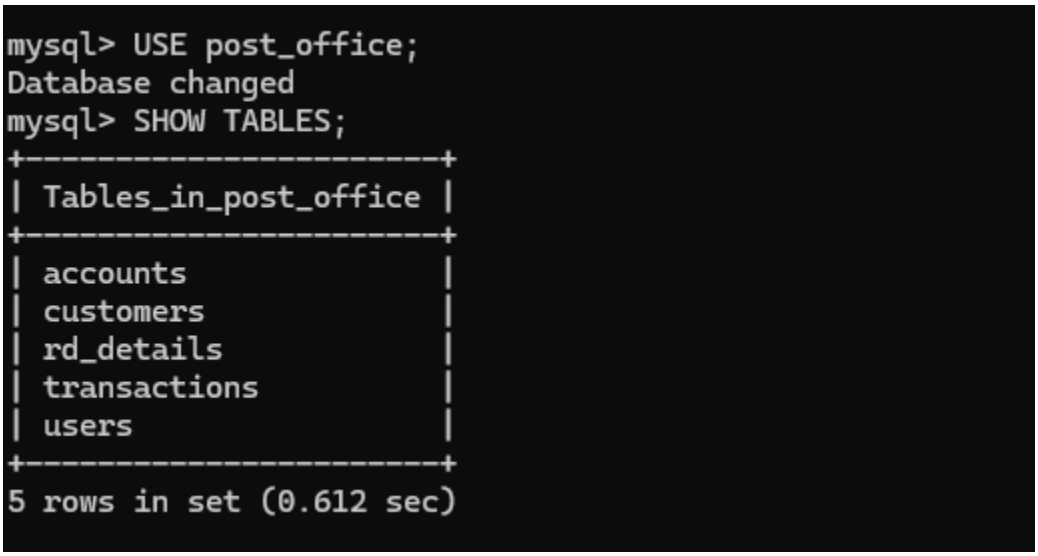
```
def rd_deposit(acc_no):

    cur.execute("SELECT monthly_amount,months_completed FROM
rd_details WHERE acc_no=%s", (acc_no,))

    monthly, completed = cur.fetchone()



    if completed < 60:

        installments = min(int(input("Installments: ")), 60-
completed)

        total = monthly * installments

        cur.execute("UPDATE accounts SET balance=balance+%s WHERE
acc_no=%s", (total, acc_no))

        cur.execute("UPDATE rd_details SET
months_completed=months_completed+%s WHERE acc_no=%s",

                    (installments, acc_no))

        print("✅ RD deposit successful")



def rd_maturity(acc_no, sb_acc):

    # Recalculate full 6.7% compound interest for all months

    monthly, months = cur.execute("SELECT
monthly_amount,months_completed FROM rd_details WHERE acc_no=%s",
(acc_no,)).fetchone()

    balance = Decimal(0)



    for m in range(months):

        balance += monthly

        interest = (balance * Decimal('6.7')) / 1200  # Monthly
compounding

        balance += interest
```

```
    # Transfer to SB + Close RD

    cur.execute("UPDATE accounts SET balance=balance+%s WHERE
acc_no=%s", (balance, sb_acc))

    cur.execute("UPDATE accounts SET status='Closed' WHERE
acc_no=%s", (acc_no,))

    con.commit()


# ================ 7. TD MATURITY ================

def td_maturity(td_acc, sb_acc):

    principal = cur.execute("SELECT balance FROM accounts WHERE
acc_no=%s", (td_acc,)).fetchone()[0]

    maturity = principal * (1 + 0.069)  # 6.9% for 12 months


    cur.execute("UPDATE accounts SET balance=balance+%s WHERE
acc_no=%s", (maturity, sb_acc))

    cur.execute("UPDATE accounts SET status='Closed' WHERE
acc_no=%s", (td_acc,))

    print(f"✅ TD maturity ₹{maturity:.2f} transferred to SB")


# ================ 8. SEARCH FUNCTION ================

def search_by_aadhaar(aadhaar):

    # Step 1: Find customer CIF

    cur.execute("SELECT customer_id FROM customers WHERE
aadhaar=%s", (aadhaar,))

    cif = cur.fetchone()


    # Step 2: Get all accounts
```

```
    cur.execute("SELECT acc_no,acc_type,balance FROM accounts
WHERE customer_id=%s", (cif[0],))

    print("All accounts for this customer:")

    for row in cur.fetchall(): print(row)


# ================ MAIN MENU ================

def main_menu():

    while True:

        print("""

1. Create Account     6. Search

2. Deposit           7. Close Account

3. Withdraw          8. Schemes Menu

4. Balance Enquiry   9. Forms

5. Interest Calc    10. Logout

0. Exit

        """)

        choice = input("Enter choice: ")


        # Route to appropriate function

        functions = { '1': create_account, '2': deposit, '3':
withdraw, ... }

        if choice in functions: functions[choice]()
```

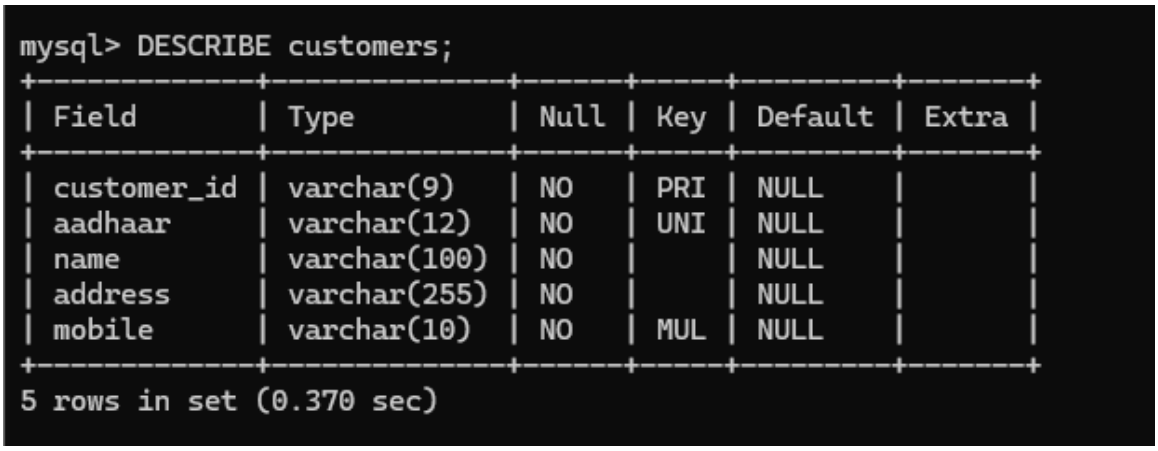
20. OUTPUT SCREENSHOTS (MYSQL)

Output Screen 1: Tables



```
mysql> USE post_office;
Database changed
mysql> SHOW TABLES;
+----------------------+
| Tables_in_post_office |
+----------------------+
| accounts             |
| customers            |
| rd_details           |
| transactions         |
| users                |
+----------------------+
5 rows in set (0.612 sec)
```

Output Screen 2: Customers Table



```
mysql> DESCRIBE customers;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| customer_id | varchar(9)  | NO   | PRI | NULL    |       |
| aadhaar     | varchar(12) | NO   | UNI | NULL    |       |
| name        | varchar(100)| NO   |     | NULL    |       |
| address     | varchar(255)| NO   |     | NULL    |       |
| mobile      | varchar(10) | NO   | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.370 sec)
```

Output Screen 3: Accounts Table

```
mysql> DESCRIBE accounts;
+---------------+-------------------------+------+-----+-------------------+-------------------+
| Field         | Type                    | Null | Key | Default           | Extra             |
+---------------+-------------------------+------+-----+-------------------+-------------------+
| acc_no        | varchar(12)             | NO   | PRI | NULL              |                   |
| name          | varchar(100)            | NO   |     | NULL              |                   |
| address       | varchar(255)            | NO   |     | NULL              |                   |
| mobile        | varchar(10)             | NO   | MUL | NULL              |                   |
| acc_type      | enum('SB','RD','TD')    | NO   | MUL | NULL              |                   |
| balance       | decimal(12,2)           | NO   |     | 0.00              |                   |
| customer_id   | varchar(9)              | NO   | MUL | NULL              |                   |
| status        | enum('Active','Closed') | NO   | MUL | Active            |                   |
| created_on    | timestamp               | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| sb_unique_key | varchar(20)             | YES  | UNI | NULL              |                   |
+---------------+-------------------------+------+-----+-------------------+-------------------+
```

Output Screen 4: DESCRIBE rd_details;

```
mysql> DESCRIBE rd_details;
+------------------+---------------+------+-----+---------+-------+
| Field            | Type          | Null | Key | Default | Extra |
+------------------+---------------+------+-----+---------+-------+
| acc_no           | varchar(12)   | NO   | PRI | NULL    |       |
| monthly_amount   | decimal(10,2) | NO   |     | NULL    |       |
| months_completed | int           | NO   |     | 0       |       |
+------------------+---------------+------+-----+---------+-------+
3 rows in set (0.022 sec)
```

Output Screen 5: DESCRIBE rd_details;

```
mysql> DESCRIBE transactions;
+----------+---------------+------+-----+-------------------+-------------------+
| Field    | Type          | Null | Key | Default           | Extra             |
+----------+---------------+------+-----+-------------------+-------------------+
| txn_id   | int           | NO   | PRI | NULL              | auto_increment    |
| acc_no   | varchar(12)   | NO   | MUL | NULL              |                   |
| txn_type | varchar(30)   | NO   | MUL | NULL              |                   |
| amount   | decimal(10,2) | NO   |     | NULL              |                   |
| txn_date | timestamp     | YES  | MUL | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+----------+---------------+------+-----+-------------------+-------------------+
5 rows in set (0.026 sec)
```

Output Screen 6: Users Table(login)

```
mysql> DESCRIBE users;
+---------------+-----------------------------------+------+-----+---------+-------+
| Field         | Type                              | Null | Key | Default | Extra |
+---------------+-----------------------------------+------+-----+---------+-------+
| username      | varchar(50)                       | NO   | PRI | NULL    |       |
| password_hash | varchar(64)                       | NO   |     | NULL    |       |
| role          | enum('POSTMASTER','CLERK')        | NO   |     | CLERK   |       |
| status        | enum('ACTIVE','INACTIVE')         | NO   |     | ACTIVE  |       |
+---------------+-----------------------------------+------+-----+---------+-------+
4 rows in set (0.021 sec)
```

Output Screen 7: Accounts

```
ysql> SELECT * FROM accounts;
+------------+---------------+-----------------------------+------------+----------+----------+-------------+--------+---------------------+---------------+
| acc_no     | name          | address                     | mobile     | acc_type | balance  | customer_id | status | created_on          | sb_unique_key |
+------------+---------------+-----------------------------+------------+----------+----------+-------------+--------+---------------------+---------------+
| 010100000001 | Anubhav Mishra | Katia Tola, Shahjahanpur   | 9119621379 | SB       |  5000.00 | 935942240   | Active | 2026-01-26 18:16:07 | 935942240     |
| 010100000002 | Saumya Mishra  | Cantt Area, Shahjahanpur   | 9621049512 | SB       |  2500.00 | 935942241   | Active | 2026-01-26 18:16:07 | 935942241     |
| 010100000003 | Neelam Devi    | Tilhar Road, Shahjahanpur  | 9839012345 | SB       |  1200.00 | 935942242   | Active | 2026-01-26 18:16:07 | 935942242     |
| 010100000004 | Rahul Verma    | Powayan, Shahjahanpur      | 9305123456 | SB       |  7000.00 | 935942243   | Active | 2026-01-26 18:16:07 | 935942243     |
| 010100000005 | Aman Singh     | Khutar, Shahjahanpur       | 9456123789 | SB       |  1500.00 | 935942244   | Active | 2026-01-26 18:16:07 | 935942244     |
| 010100000006 | Pooja Gupta    | Katra, Shahjahanpur        | 9145123678 | SB       |  3200.00 | 935942245   | Active | 2026-01-26 18:16:07 | 935942245     |
| 010100000007 | Rakesh Kumar   | Jalalabad, Shahjahanpur    | 9876501234 | SB       |  1800.00 | 935942246   | Active | 2026-01-26 18:16:07 | 935942246     |
| 010100000008 | Shivani Sharma | Mirzapur, Shahjahanpur     | 9800012345 | SB       |  5100.00 | 935942247   | Active | 2026-01-26 18:16:07 | 935942247     |
| 010100000009 | Karan Yadav    | Nigohi, Shahjahanpur       | 9123456780 | SB       |  2400.00 | 935942248   | Active | 2026-01-26 18:16:07 | 935942248     |
| 010100000010 | Mohit Jain     | Sadar Bazar, Shahjahanpur  | 9012345678 | SB       |  6000.00 | 935942249   | Active | 2026-01-26 18:16:07 | 935942249     |
| 010100000011 | Sakshi Patel   | Banda Road, Shahjahanpur   | 9898989898 | SB       |   800.00 | 935942250   | Active | 2026-01-26 18:16:07 | 935942250     |
| 010100000012 | Vikas Tiwari   | Civil Lines, Shahjahanpur  | 9797979797 | SB       |   900.00 | 935942251   | Active | 2026-01-26 18:16:07 | 935942251     |
| 010100000013 | Riya Singh     | Shahjahanpur City          | 9696969696 | SB       |   500.00 | 935942252   | Active | 2026-01-26 18:16:07 | 935942252     |
| 010100000014 | Arjun Saxena   | Khari Baoli, Shahjahanpur  | 9595959595 | SB       |  4500.00 | 935942253   | Active | 2026-01-26 18:16:07 | 935942253     |
| 010100000015 | Sneha Gupta    | Clock Tower, Shahjahanpur  | 9494949494 | SB       |  2000.00 | 935942254   | Active | 2026-01-26 18:16:07 | 935942254     |
| 010205181655 | somesh         | devkali                    | 9119621379 | SB       | 70539.73 | 815486247   | Active | 2026-01-29 21:42:48 | 815486247     |
| 010565625076 | saumya         | kati                       | 1122334455 | SB       | 87521.58 | 144131371   | Active | 2026-01-26 17:45:26 | 144131371     |
| 020200000001 | Anubhav Mishra | Katia Tola, Shahjahanpur   | 9119621379 | RD       |  1000.00 | 935942240   | Active | 2026-01-26 18:16:07 | NULL          |
| 020200000002 | Saumya Mishra  | Cantt Area, Shahjahanpur   | 9621049512 | RD       |   500.00 | 935942241   | Active | 2026-01-26 18:16:07 | NULL          |
| 020200000003 | Rahul Verma    | Powayan, Shahjahanpur      | 9305123456 | RD       |  2000.00 | 935942243   | Active | 2026-01-26 18:16:07 | NULL          |
| 020200000004 | Rahul Verma    | Powayan, Shahjahanpur      | 9305123456 | RD       |  1500.00 | 935942243   | Active | 2026-01-26 18:16:07 | NULL          |
| 020257516469 | saumya         | kati                       | 1122334455 | RD       |     0.00 | 144131371   | Closed | 2026-01-26 17:46:20 | NULL          |
| 020629569589 | saumya         | kati                       | 1122334455 | RD       |  1000.00 | 144131371   | Active | 2026-01-26 17:46:52 | NULL          |
| 020906702098 | somesh         | devkali                    | 9119621379 | RD       |     0.00 | 815486247   | Closed | 2026-01-29 21:47:41 | NULL          |
| 030199388376 | saumya         | kati                       | 1122334455 | TD       |  1000.00 | 144131371   | Active | 2026-01-26 17:47:45 | NULL          |
| 030300000001 | Anubhav Mishra | Katia Tola, Shahjahanpur   | 9119621379 | TD       | 10000.00 | 935942240   | Active | 2026-01-26 18:16:07 | NULL          |
| 030300000002 | Saumya Mishra  | Cantt Area, Shahjahanpur   | 9621049512 | TD       |  5000.00 | 935942241   | Active | 2026-01-26 18:16:07 | NULL          |
| 030300000003 | Sneha Gupta    | Clock Tower, Shahjahanpur  | 9494949494 | TD       |  8000.00 | 935942254   | Active | 2026-01-26 18:16:07 | NULL          |
| 030410056081 | saumya         | kati                       | 1122334455 | TD       |  1000.00 | 144131371   | Active | 2026-01-26 17:47:26 | NULL          |
+------------+---------------+-----------------------------+------------+----------+----------+-------------+--------+---------------------+---------------+
29 rows in set (0.079 sec)
```

Output Screen 8: View RD Installments

```
mysql> SELECT * FROM rd_details;
+--------------+----------------+-------------------+
| acc_no       | monthly_amount | months_completed  |
+--------------+----------------+-------------------+
| 020200000001 |        1000.00 |                12 |
| 020200000002 |         500.00 |                 6 |
| 020200000003 |        2000.00 |                24 |
| 020200000004 |        1500.00 |                36 |
| 020257516469 |        1000.00 |                60 |
| 020629569589 |        1000.00 |                 1 |
| 020906702098 |        1000.00 |                59 |
+--------------+----------------+-------------------+
```

Output Screen 5: View Transactions

```
mysql> SELECT * FROM transactions;
+--------+--------------+--------------+----------+---------------------+
| txn_id | acc_no       | txn_type     | amount   | txn_date            |
+--------+--------------+--------------+----------+---------------------+
|      1 | 010100000001 | OPENING      |  5000.00 | 2026-01-26 18:16:07 |
|      2 | 010100000002 | OPENING      |  2500.00 | 2026-01-26 18:16:07 |
|      3 | 010100000003 | OPENING      |  1200.00 | 2026-01-26 18:16:07 |
|      4 | 010100000004 | OPENING      |  7000.00 | 2026-01-26 18:16:07 |
|      5 | 010100000005 | OPENING      |  1500.00 | 2026-01-26 18:16:07 |
|      6 | 010100000001 | DEPOSIT      |  1000.00 | 2026-01-26 18:16:07 |
|      7 | 010100000001 | WITHDRAW     |   500.00 | 2026-01-26 18:16:07 |
|      8 | 010100000002 | DEPOSIT      |   200.00 | 2026-01-26 18:16:07 |
|      9 | 010100000003 | WITHDRAW     |   100.00 | 2026-01-26 18:16:07 |
|     10 | 010100000004 | DEPOSIT      |  1500.00 | 2026-01-26 18:16:07 |
|     11 | 020200000001 | RD_OPEN      |  1000.00 | 2026-01-26 18:16:07 |
|     12 | 020200000002 | RD_OPEN      |   500.00 | 2026-01-26 18:16:07 |
|     13 | 020200000003 | RD_OPEN      |  2000.00 | 2026-01-26 18:16:07 |
|     14 | 020200000004 | RD_OPEN      |  1500.00 | 2026-01-26 18:16:07 |
|     15 | 020200000001 | RD_DEPOSIT   |  1000.00 | 2026-01-26 18:16:07 |
|     16 | 020200000001 | RD_DEPOSIT   |  1000.00 | 2026-01-26 18:16:07 |
|     17 | 020200000001 | RD_DEPOSIT   |  1000.00 | 2026-01-26 18:16:07 |
|     18 | 020200000002 | RD_DEPOSIT   |   500.00 | 2026-01-26 18:16:07 |
|     19 | 020200000003 | RD_DEPOSIT   |  2000.00 | 2026-01-26 18:16:07 |
|     20 | 030300000001 | TD_OPEN      | 10000.00 | 2026-01-26 18:16:07 |
|     21 | 030300000002 | TD_OPEN      |  5000.00 | 2026-01-26 18:16:07 |
|     22 | 030300000003 | TD_OPEN      |  8000.00 | 2026-01-26 18:16:07 |
|     23 | 010100000006 | DEPOSIT      |  1000.00 | 2026-01-26 18:16:07 |
|     24 | 010100000007 | WITHDRAW     |   200.00 | 2026-01-26 18:16:07 |
|     25 | 010100000008 | DEPOSIT      |   700.00 | 2026-01-26 18:16:07 |
|     26 | 010100000009 | DEPOSIT      |   900.00 | 2026-01-26 18:16:07 |
|     27 | 010100000010 | WITHDRAW     |   300.00 | 2026-01-26 18:16:07 |
|     28 | 010100000011 | DEPOSIT      |   500.00 | 2026-01-26 18:16:07 |
|     29 | 010100000012 | DEPOSIT      |   700.00 | 2026-01-26 18:16:07 |
|     30 | 010100000013 | WITHDRAW     |   100.00 | 2026-01-26 18:16:07 |
|     31 | 010100000014 | DEPOSIT      |  1200.00 | 2026-01-26 18:16:07 |
|     32 | 010100000015 | DEPOSIT      |  2000.00 | 2026-01-26 18:16:07 |
|     33 | 010100000001 | INTEREST     |   120.00 | 2026-01-26 18:16:07 |
|     34 | 010100000002 | INTEREST     |    60.00 | 2026-01-26 18:16:07 |
|     35 | 010100000003 | INTEREST     |    45.00 | 2026-01-26 18:16:07 |
|     36 | 010100000004 | INTEREST     |   150.00 | 2026-01-26 18:16:07 |
|     37 | 020200000004 | RD_PREMATURE |  5000.00 | 2026-01-26 18:16:07 |
|     38 | 020200000003 | RD_DEPOSIT   |  2000.00 | 2026-01-26 18:16:07 |
|     39 | 020200000003 | RD_DEPOSIT   |  2000.00 | 2026-01-26 18:16:07 |
|     40 | 020200000003 | RD_DEPOSIT   |  2000.00 | 2026-01-26 18:16:07 |
|     41 | 030300000001 | TD_INTEREST  |   690.00 | 2026-01-26 18:16:07 |
|     42 | 030300000002 | TD_INTEREST  |   345.00 | 2026-01-26 18:16:07 |
|     43 | 030300000003 | TD_INTEREST  |   552.00 | 2026-01-26 18:16:07 |
+--------+--------------+--------------+----------+---------------------+
```

21. OUTPUT SCREENSHOTS (CLI INTERFACE)

Output Screenshot 1: Main menu after login

```
=== POST OFFICE LOGIN ===
Username: postmaster
Password: 1234

☑ Login Successful! Role: POSTMASTER

=== POST OFFICE ACCOUNT SYSTEM ===
1. Open Account
2. Deposit
3. Withdraw
4. Balance Enquiry
5. Interest Calculation
6. Search Account
7. Close Account
8. Schemes Menu
9. Post Office Forms
10. Logout
0. Exit

Enter Choice:
```

Output Screenshot 2: Creating New Account

```
Enter Choice: 1

=== CREATE NEW ACCOUNT ===

=== CUSTOMER IDENTIFICATION ===
Enter Customer ID (10 digits) OR press Enter for NEW customer:

🆕 New Customer Registration
Customer Name: Adarsh
Customer Address: Anandpuram Colony Spn
Customer Mobile (10 digits): 9119621379
Customer Aadhaar (12 digits): 224466880022

☑ New Customer Created Successfully!
Customer ID (CIF): 105598640
Account Type (SB / RD / TD): sb
Opening Balance (Min 500): 500

☑ Account Created Successfully!
Customer ID (CIF): 105598640
Account Number: 010973975273
Account Type: SB
```

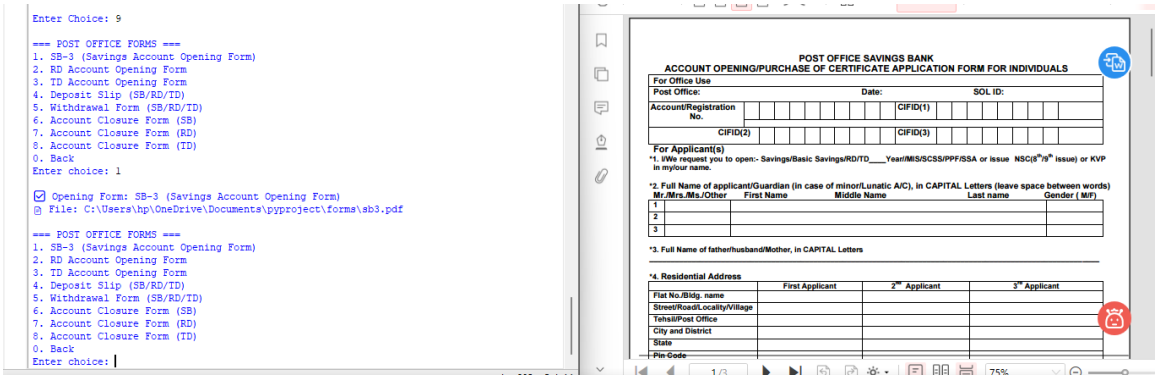
Output Screenshot 3: Searching Account

```
Enter Choice: 6

--- SEARCH ACCOUNT ---
1. Search by Name
2. Search by Account Number
3. Search by Mobile Number
4. Search by Aadhaar Number ☑ (NEW)
0. Back

Enter Choice: 1
Enter Name: Adarsh

ACC NO | NAME | TYPE | BALANCE | STATUS
----------------------------------------
('010375294773', 'Adarsh', 'SB', Decimal('500.00'), 'Active')
('010973975273', 'Adarsh', 'SB', Decimal('500.00'), 'Active')
```

Output Screenshot 4: RD deposit

```
Enter Choice: 8

--- SCHEMES ---
1. RD Scheme
2. SB Scheme
3. TD Scheme
0. Back

Choice: 1

--- RD SCHEME ---
1. Monthly RD Deposit
2. RD Compound Interest
3. RD Schedule
4. RD Maturity Transfer
5. RD Scheme Info ☑
0. Back

Choice: 1
RD Account Number: 020200000002
Enter number of installments to deposit: 1

☑ RD Deposit Successful
Installments Deposited: 1
Amount Deposited: 500.0
Total Installments Completed: 7
```

Output Screenshot 5: P.O. Forms



22. Advantages of The Project

1. **Reduces manual work**
   The system automates post office account management, reducing paperwork and manual record keeping.
2. **Fast and accurate operations**
   All calculations such as balance updates and interest calculation are performed automatically, reducing chances of human error.
3. **User-friendly menu-driven interface**
   The menu-driven design makes the system easy to use even for beginners.
4. **Efficient data management**
   The use of MySQL database ensures that account records are stored securely and can be retrieved easily.
5. **Scheme-wise organization**
   Separate handling of SB, RD, and TD schemes improves clarity and better understanding of different account types.
6. **Input validation and data security**
   Validation checks such as mobile number length and minimum balance rules help maintain correct and reliable data.
7. **Realistic banking simulation**
   The project closely simulates real post office banking operations like RD monthly deposits and maturity transfers.
8. **Easy maintenance and scalability**
   The modular design allows future enhancements without major changes in the existing code.
9. **Educational value**
   The project helps in understanding Python programming, SQL queries, and database connectivity practically.
10. **Cost-effective solution**
    Since the project uses open-source tools, it is economical and easy to implement.

23. Limitations and Future Enhancements

23.1 Current Limitations

- No transaction history logging
- No email notifications for transactions
- No multi-currency support
- Fixed interest rates (no dynamic updates)
- No overdraft facility
- RD schedule uses different rate (5.8%) than maturity calculation (6.7%)

23.2 Recommended Enhancements

- Implement transaction audit trail table
- Add email/SMS notification system
- Integrate with online payment gateways
- Create admin dashboard for monitoring
- Add document upload for KYC
- Implement interest rate management module
- Add FD (Fixed Deposit) scheme
- Create mobile application layer

---

24. Database Backup and Recovery

24.1 Data Integrity

- All write operations use `con.commit()` for persistent storage
- Failed operations use `con.rollback()` to prevent partial updates
- Connection timeout set to 5 seconds (prevents hanging)

24.2 Recommended Backup Strategy

- Daily full database backup
- Monthly incremental backups
- Store backups in separate location from primary server

- Test restore procedures quarterly

- **Scalability**: Customer-centric CIF model allowing multiple accounts

The system successfully demonstrates core banking operations while maintaining compliance with postal banking scheme guidelines for interest rates, tenures, and operational constraints.

---

25. Conclusion

The Post Office Account Management System provides a comprehensive solution for managing three distinct banking schemes with emphasis on:

- **Security**: Password hashing, SQL injection prevention
- **Accuracy**: Decimal-based calculations, compound interest formulas
- **User Experience**: Hierarchical menu navigation, input validation
- **Data Integrity**: Transaction management, rollback mechanisms

26. References

The following resources were referred to during the development of this project:

1. Python Official Documentation
   https://docs.python.org
2. MySQL Official Documentation
   https://dev.mysql.com/doc
3. mysql-connector-python Documentation
   https://dev.mysql.com/doc/connector-python/en
4. CBSE Computer Science Curriculum (Class XII)
5. Online tutorials and learning resources for Python and MySQL