Christ University

Department of Computer Science and Engineering

Kengeri Campus, Bangalore

BANKMATE

-Bank management system-

Couse and code: Computer Programming (CSE134P)

Group members:

1. Ryan Thomas Philips → 2562163
2. Hrishikesh.P.S → 2562142
3. Jovian Pereira → 2562147

ACKNOWLEDGEMENT

I solemnly take this opportunity to thank all the helping hands who supported me in completing this project. First of all, I thank the Almighty for keeping me hale and healthy, enabling me to successfully complete my work.

I wish to express my sincere gratitude to **Mr. Bijeesh T V** , Teacher in computer programming (CSE134P) at Christ University (Deemed to be University), Kengeri Campus , for his valuable guidance, encouragement, and continuous support throughout the completion of this project. I also extend my appreciation to my team members for their cooperation and collective efforts in bringing this project to fruition.

Last but not least, I express my heartfelt thanks to my loving parents and friends for their prayers, suggestions, and constant encouragement that helped me complete this project successfully.

CONTENTS

ABSTRACT

C is a powerful and structured programming language widely used for developing system software and application software. It provides efficient control over system resources, making it suitable for building fast and reliable applications. Its simplicity, portability, and rich set of library functions make it an ideal choice for implementing fundamental concepts of programming, data structures, and file handling.

This project titled **"BankMate – Bank Management System"** is developed using the C programming language to demonstrate the application of fundamental programming concepts in a real-world scenario. The system allows users to create and manage customer accounts, store essential details, and perform basic banking operations such as deposits, withdrawals, balance inquiries, and account updates. It incorporates the use of an embedded database for managing customer records, ensuring secure & organized data storage. To enhance data integrity and security, the system implements the Verhoeff algorithm for validating Aadhaar numbers and uses bitwise operations for PIN encryption, preventing direct exposure of sensitive credentials. These features collectively make the project a secure, reliable, and efficient banking solution prototype.

The primary objective of this project is to provide a simple yet efficient platform for understanding how banking transactions can be simulated using C. It helps improve logical thinking, problem-solving skills, and understanding of concepts like file handling, conditional statements, loops, functions, and database integration. This project can serve as a foundation for developing more advanced banking systems with enhanced features such as encryption, GUI interfaces, and network-based operations.

SYSTEM REQUIREMENTS

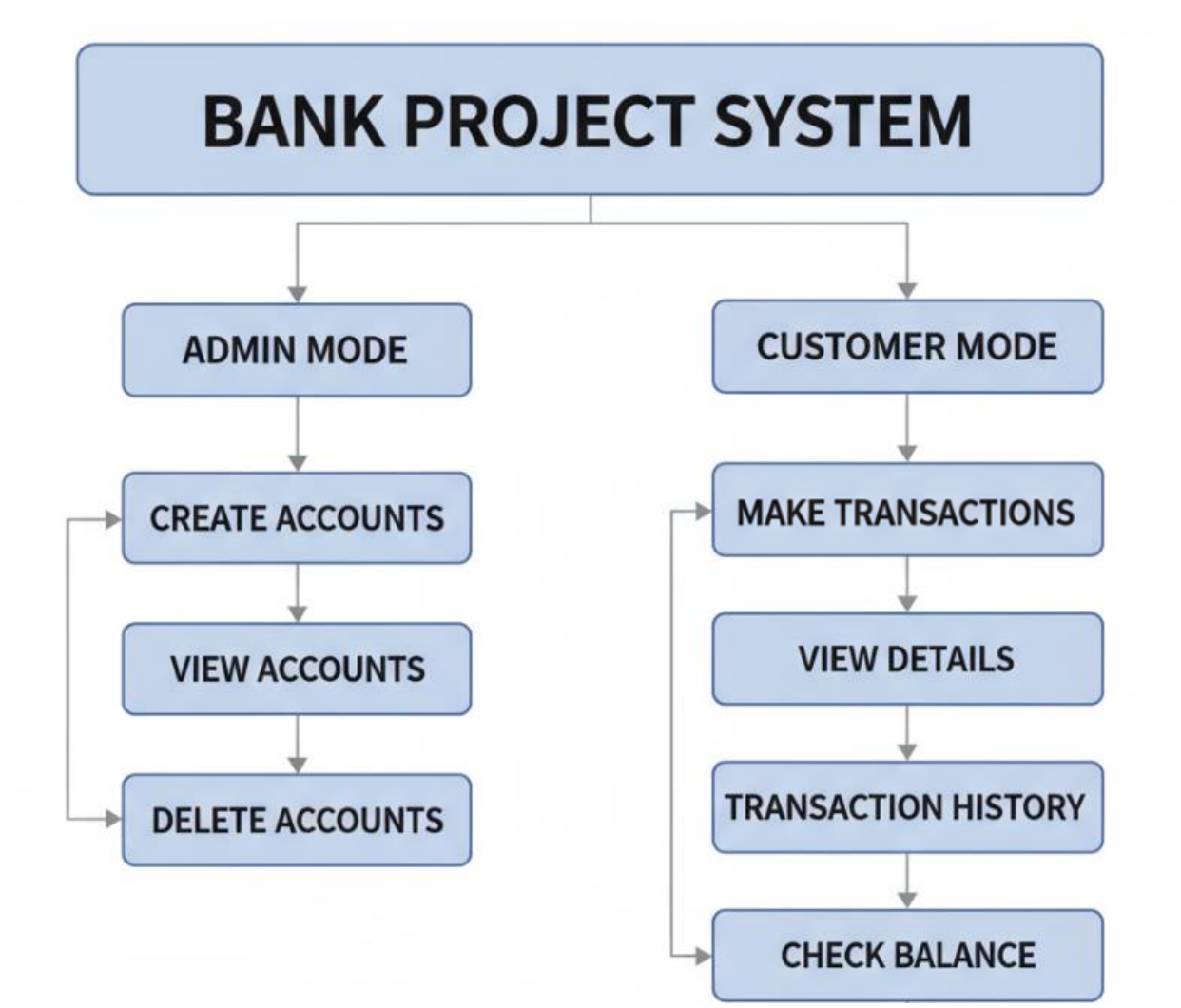
HARDWARE COMPONENTS:

- **RAM**: 2 GB (Minimum) / 4 GB (Recommended)
- **Operating System**: 32-bit x86 / 64-bit x64 (Recommended)
- **Hard Disk Memory**: Minimum 250 MB free
- **Processor**: Dual Core 2.80 GHz or greater
- **Screen Resolution**: 1366 × 768 (Optimal)
- **Graphics Card**: Minimum 64 MB (basic integrated graphics sufficient)

SOFTWARE COMPONENTS:

- **Platform**: windows 7 / 8 / 10 / 11 with latest updates
- **Programming Language**: C
- **Compiler**: / GCC / MinGW / Code::Blocks / Dev-C++
- **Database**: SQLite (embedded database library)
- **Editor/IDE**: Any C-compatible IDE or text editor

Project Design



Databases

```
D:\Ryan\projects\BankMate>sqlite3 bankdb.db
SQLite version 3.50.4 2025-07-30 19:33:53
Enter ".help" for usage hints.
sqlite> .tables
admins        customers     transactions
sqlite> .schema
CREATE TABLE admins (
    admin_id    INTEGER PRIMARY KEY AUTOINCREMENT,
    username    TEXT UNIQUE NOT NULL,
    password    TEXT NOT NULL
);
CREATE TABLE sqlite_sequence(name,seq);
CREATE TABLE transactions (
    txn_id           INTEGER PRIMARY KEY AUTOINCREMENT,
    from_account_no  INTEGER NOT NULL,
    to_account_no    INTEGER NOT NULL,
    amount           DECIMAL(15,2) NOT NULL,
    txn_time         TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    remarks          TEXT,
    FOREIGN KEY(from_account_no) REFERENCES customers(account_no),
    FOREIGN KEY(to_account_no)   REFERENCES customers(account_no)
);
CREATE TABLE customers (
    account_no     INTEGER PRIMARY KEY,
    name            TEXT NOT NULL,
    dob             TEXT NOT NULL,
    aadhaar         CHAR(12) UNIQUE NOT NULL,
    mobile           CHAR(10) NOT NULL,
    email            TEXT,
    address          TEXT,
    account_type     TEXT CHECK(account_type IN ('SAVINGS','CURRENT')) NOT NULL,
    balance           DECIMAL(15,2) DEFAULT 0.00,
    pin                CHAR(6) NOT NULL
);
sqlite>
```

bankmate.c

SOURCE CODE

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#include <conio.h>
#include "sqlite/sqlite3.c"
#include "sqlite/sqlite3.h"

sqlite3 *db;
sqlite3_stmt *stmt;
int rc;

int choice();
int admin_options();
void admin();
void create_acc();
void customer_details();
void delete_acc();
void logout();
void customers();
void get_pin(char *pin, int max_len);
void to_uppercase(char *str);

//=======================================START PAGE==========================================
int choice() {
    int option;
    printf("      _____________________________________________________________________________\n");
    printf("     |                                                                             |\n");
    printf("     |                          1-ADMIN LOGIN                                      |\n");
    printf("     |                          2-CUSTOMER LOGIN                                   |\n");
    printf("     |                          3-EXIT                                             |\n");
    printf("     |_____________________________________________________________-_____________|\n");
    printf("                                ENTER CHOICE:");
    scanf("%d", &option);
    printf("     |_____________________________________________________________-_____________|\n\n");

    if (option<4 && option>0)
        return option;
    else
        return choice();
```

```c
}

//==================================ADMIN=====================================================
int admin_option(){
    int option;
    printf("\n     |  1-CREATE BANK ACCOUNT
|\n");
    printf("     |  2-CUSTOMER ACCOUNT DETAILS
|\n");
    printf("     |  3-DELETE BANK ACCOUNT
|\n");
    printf("     |  4-LOG OUT
|\n");
    printf("     |_____________________________________________________-
___________|\n");
    printf("     |      ENTER CHOICE:");
    scanf("%d", &option);
    printf("     |_____________________________________________________-
___________|\n\n");

    if (option<5 && option>0){
        return option;}
    else{
        printf("    INVALID INPUT\n\n");
        return admin_option();
    }
}

void admin() {
    char user[50], pswrd[100];
    int option;
    printf("      _____________________________________________________-
___________\n");
    printf("     |
|\n");
    printf("     |                         ADMIN LOGIN PANEL
|\n");
    printf("     |_____________________________________________________-
___________|\n");
    printf("     |
|\n");
    printf("     |  USERNAME : "); scanf("%s", user);
    printf("     |  PASSWORD : "); scanf("%s", pswrd);
    printf("     |_____________________________________________________-
___________|\n");


    // Prepare SQL query to fetch admin by username and password
    const char *sql = "SELECT admin_id, username, password FROM admins WHERE username=? AND
password=?;";

    rc = sqlite3_prepare_v2(db, sql, -1, &stmt, 0);
```

```
    if (rc != SQLITE_OK) {
        fprintf(stderr, "Failed to prepare statement: %s\n", sqlite3_errmsg(db));
        sqlite3_close(db);
        return;
    }

    // Bind user inputs to the SQL statement
    sqlite3_bind_text(stmt, 1, user, -1, SQLITE_TRANSIENT);
    sqlite3_bind_text(stmt, 2, pswrd, -1, SQLITE_TRANSIENT);


    // Execute the query
    if (sqlite3_step(stmt) == SQLITE_ROW) {
        int admin_id = sqlite3_column_int(stmt, 0);
        const unsigned char *uname = sqlite3_column_text(stmt, 1);
        const unsigned char *pwd = sqlite3_column_text(stmt, 2);

        printf("     |
|\n");
        printf("     |  Login Successful!\n");
        printf("     |  Admin ID   : %d
|\n", admin_id);
    } else {
        printf("     |  Invalid username or password.
|\n");
        admin();
    }
    sqlite3_finalize(stmt);

    printf("     |  Welcome, %s! \n", user);
    option=admin_option();
    switch (option) {
        case 1: create_acc(); break;
        case 2: customer_details(); break;
        case 3: delete_acc(); break;
        case 4: logout(); break;
        default: break;
    }
}


//===========view customer details====================
void view_details(const char *search_name){
    char upper_name[50];
    strcpy(upper_name, search_name);
    to_uppercase(upper_name);
    const char *sql = "SELECT account_no, name, dob, aadhaar, mobile, email, address,
account_type, balance "
                      "FROM customers WHERE name = ?;";

    int rc = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
```

```
    if (rc != SQLITE_OK) {
        fprintf(stderr, "Failed to prepare statement: %s\n", sqlite3_errmsg(db));
        sqlite3_close(db);
        return;
    }

    // Bind the name parameter to the SQL statement
    sqlite3_bind_text(stmt, 1, upper_name, -1, SQLITE_STATIC);

    while ((rc = sqlite3_step(stmt)) == SQLITE_ROW) {
        int accno = sqlite3_column_int(stmt, 0);
        const unsigned char *name = sqlite3_column_text(stmt, 1);
        const unsigned char *dob = sqlite3_column_text(stmt, 2);
        const unsigned char *aadhaar = sqlite3_column_text(stmt, 3);
        const unsigned char *mobile = sqlite3_column_text(stmt, 4);
        const unsigned char *email = sqlite3_column_text(stmt, 5);
        const unsigned char *address = sqlite3_column_text(stmt, 6);
        const unsigned char *account_type = sqlite3_column_text(stmt, 7);
        double balance = sqlite3_column_double(stmt, 8);

        int len = strlen(aadhaar);
        int i, j = 0;
        char output[20];
        for (i = 0; i < len; i++) {
            if (i < len - 4) {
                output[j++] = 'X';
            } else {
                output[j++] = aadhaar[i];
            }
            // Add space after every 4 characters (except at the very end)
            if ((i + 1) % 4 == 0 && i != len - 1) {
                output[j++] = ' ';
            }
        }
        output[j] = '\0';

        printf("     |  ACCOUNT NO.     : %lld\n", accno);
        printf("     |  NAME            : %s\n", name);
        printf("     |  DATE OF BIRTH   : %s\n", dob);
        printf("     |  AADHAR NUMBER   : %s\n", output);
        printf("     |  MOBILE NUMBER   : +91 %s\n", mobile);
        printf("     |  EMAIL           : %s\n", email);
        printf("     |  ADDRESS         : %s\n", address);
        printf("     |  BALANCE         : %.2f\n", balance);
        printf("     |_____________________________________________________-
___________|\n\n");
    }

    if (rc != SQLITE_DONE) {
        fprintf(stderr, "SELECT error: %s\n", sqlite3_errmsg(db));
```

```
    }
    sqlite3_finalize(stmt);
}

//================================CUSTOMER=================================================
void customers() {
    char name[50], pin[10], db_pin[20];
    long long acc_no;
    int login_success = 0, attempts = 0;
    printf("      ____________________________________________________________________
\n");
    printf("     |
|\n");
    printf("     |                          CUSTOMER LOGIN PANEL
|\n");
    printf("     |_____________________________________________________________-
__________|\n");

    while (!login_success && attempts < 3) {
        printf("     |  NAME         : "); scanf(" %[^\n]", name);
        printf("     |  ACCOUNT NO.  : "); scanf(" %lld", &acc_no);
        printf("     |  PIN          : "); get_pin(pin, sizeof(pin));
        char upper_name[50];
        strcpy(upper_name, name);
        to_uppercase(upper_name);

        const char *sql = "SELECT pin FROM customers WHERE name=? AND account_no=?;";
        rc = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
        if (rc != SQLITE_OK) { printf("DB error: %s\n", sqlite3_errmsg(db)); return; }
        sqlite3_bind_text(stmt, 1, upper_name, -1, SQLITE_TRANSIENT);
        sqlite3_bind_int64(stmt, 2, acc_no);

        if (sqlite3_step(stmt) == SQLITE_ROW) {
            strcpy(db_pin, (const char*)sqlite3_column_text(stmt, 0));
            int stored_pin = atoi(db_pin) >> 3;
            if (atoi(pin) == stored_pin) {
                login_success = 1;
            } else {
                printf("     |  Incorrect PIN. Try again.
|\n");
            }
        } else {
            printf("     |  Invalid name or account number.
|\n");
        }
        sqlite3_finalize(stmt);
        attempts++;
    }

    if (!login_success){
        printf("     |  Login failed.\n");
```

```c
        return;
    }

    int cust_menu = 0;
    printf("     |  Login successful! Welcome, %s!\n\n", name);
    view_details(name);

    while (1) {
        printf("      _____________________________________________________________________________\n");
        printf("     |                                                                             |\n");
        printf("     |  1-View Last 10 Transactions                                                |\n");
        printf("     |  2-View Balance                                                             |\n");
        printf("     |  3-Make Transaction                                                         |\n");
        printf("     |  4-Logout                                                                   |\n");
        printf("     |  Enter choice: ");     scanf("%d", &cust_menu);
        printf("     |_____________________________________________________________-___________|\n");

        if (cust_menu < 1 || cust_menu > 4) {
            printf("     |  Invalid choice. Try again.                                                 |\n");
            continue;
        }

        // View last 10 transactions
        if (cust_menu == 1) {
            // Join with customers table to get names for both sender and receiver
            const char *sql = "SELECT t.txn_time, c1.name as from_name, c2.name as to_name, t.amount, t.remarks "
                              "FROM transactions t "
                              "LEFT JOIN customers c1 ON t.from_account_no = c1.account_no "
                              "LEFT JOIN customers c2 ON t.to_account_no = c2.account_no "
                              "WHERE t.from_account_no=? OR t.to_account_no=? "
                              "ORDER BY t.txn_time DESC LIMIT 10;";
            rc = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);

            if (rc != SQLITE_OK){
                printf("DB error: %s\n", sqlite3_errmsg(db));
                continue;
            }
            sqlite3_bind_int64(stmt, 1, acc_no);
            sqlite3_bind_int64(stmt, 2, acc_no);
            printf("\n     _____________________________________________________________-_____________________________________\n");
```

```
262             printf("    |  Date/Time           |  From                |  To                  |  Amount   |  Remarks            |\n");
263             printf("    |----------------------|----------------------|----------------------|-----------|---------------------|\n");
264             while (sqlite3_step(stmt) == SQLITE_ROW) {
265                 const char *dt = (const char*)sqlite3_column_text(stmt, 0);
266                 const char *from_name = (const char*)sqlite3_column_text(stmt, 1);
267                 const char *to_name = (const char*)sqlite3_column_text(stmt, 2);
268                 double amt = sqlite3_column_double(stmt, 3);
269                 const char *remarks = (const char*)sqlite3_column_text(stmt, 4);
270                 printf("    | %-20s | %-20s | %-20s | %-8.2f  | %s\n", dt, from_name ? from_name : "-", to_name ? to_name : "-", amt, remarks ? remarks : "");
271             }
272             sqlite3_finalize(stmt);
273             printf("    |_____________________________________________________________________________________|\n\n");
274 
275 
276         // ======================View balance====================
277         }else if (cust_menu == 2){
278             const char *sql = "SELECT balance FROM customers WHERE account_no=?;";
279             rc = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
280 
281             if (rc != SQLITE_OK){
282                 printf("DB error: %s\n", sqlite3_errmsg(db));
283                 continue;
284             }
285             sqlite3_bind_int64(stmt, 1, acc_no);
286             if (sqlite3_step(stmt) == SQLITE_ROW){
287                 double bal = sqlite3_column_double(stmt, 0);
288                 printf("\n     |  Current Balance: %.2f\n", bal);
289             }
290             sqlite3_finalize(stmt);
291 
292 
293         // ======================Make transaction============================
294         } else if (cust_menu == 3) {
295             char recv_name[50], recv_name_up[50], recv_last4[10];
296             long long recv_acc = 0;
297             double amount = 0.0, sender_bal = 0.0;
298             double recv_bal = 0.0;
299             char pin_check[10];
300 
301             printf("     |  Enter receiver's name: "); scanf(" %[^\n]", recv_name);
302             strcpy(recv_name_up, recv_name); to_uppercase(recv_name_up);
303 
304             printf("     |  Enter last 4 digits of receiver's account: "); scanf("%s", recv_last4);
305             if (strlen(recv_last4) != 4 || !isdigit(recv_last4[0]) || !isdigit(recv_last4[1]) || !isdigit(recv_last4[2]) || !isdigit(recv_last4[3])) {
306                 printf("     |  Invalid input. Must be exactly 4 digits.\n");
```

```
307                 continue;
308             }
309
310             // Find receiver's account
311             const char *sql = "SELECT account_no, balance FROM customers WHERE name=?;";
312             rc = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
313
314             if (rc != SQLITE_OK){
315                 printf("DB error: %s\n", sqlite3_errmsg(db));
316                 continue;
317             }
318             sqlite3_bind_text(stmt, 1, recv_name_up, -1, SQLITE_TRANSIENT);
319             int found = 0;
320             while (sqlite3_step(stmt) == SQLITE_ROW) {
321                 long long acc = sqlite3_column_int64(stmt, 0);
322                 char acc_str[20]; sprintf(acc_str, "%lld", acc);
323                 int len = strlen(acc_str);
324                 if (len >= 4 && strcmp(acc_str + len - 4, recv_last4) == 0) {
325                     recv_acc = acc;
326                     recv_bal = sqlite3_column_double(stmt, 1);
327                     found = 1;
328                     break;
329                 }
330             }
331
332             sqlite3_finalize(stmt);
333             if (!found){
334                 printf("     |  Receiver not found.\n");
335                 continue;
336             }
337             if (recv_acc == acc_no) {
338                 printf("     |  Cannot transfer to the same account.\n");
339                 continue;
340             }
341
342             // Get sender balance
343             sql = "SELECT balance FROM customers WHERE account_no=?;";
344             rc = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
345
346             if (rc != SQLITE_OK){
347                 printf("DB error: %s\n", sqlite3_errmsg(db));
348                 continue;
349             }
350             sqlite3_bind_int64(stmt, 1, acc_no);
351             if (sqlite3_step(stmt) == SQLITE_ROW) {
352                 sender_bal = sqlite3_column_double(stmt, 0);
353             }
354             sqlite3_finalize(stmt);
355
356             printf("     |  Enter amount to send: "); scanf("%lf", &amount);
```

```c
            if (amount <= 0 || amount > sender_bal){
                printf("     |  Insufficient balance or invalid amount.\n");
                continue;
            }
            printf("     |  Enter your PIN to confirm: ");
            get_pin(pin_check, sizeof(pin_check));
            int stored_pin = atoi(db_pin) >> 3;
            if (atoi(pin_check) != stored_pin){
                printf("     |  Incorrect PIN. Transaction cancelled.\n");
                continue;
            }


            // Perform transaction
            printf("\n     |  Transferring %.2f to %s (Acc: %lld)...\n", amount, recv_name,
recv_acc);

            // Deduct from sender
            sql = "UPDATE customers SET balance=balance-? WHERE account_no=?;";
            rc = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
            if (rc != SQLITE_OK){
                printf("DB error: %s\n", sqlite3_errmsg(db));
                continue;
            }
            sqlite3_bind_double(stmt, 1, amount);
            sqlite3_bind_int64(stmt, 2, acc_no);
            if (sqlite3_step(stmt) != SQLITE_DONE){
                printf("     |  Error updating sender balance.\n");
                sqlite3_finalize(stmt);
                continue;
            }
            sqlite3_finalize(stmt);

            // Add to receiver
            sql = "UPDATE customers SET balance=balance+? WHERE account_no=?;";
            rc = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
            if (rc != SQLITE_OK) {
                printf("DB error: %s\n", sqlite3_errmsg(db));
                continue;
            }
            sqlite3_bind_double(stmt, 1, amount);
            sqlite3_bind_int64(stmt, 2, recv_acc);
            if (sqlite3_step(stmt) != SQLITE_DONE) {
                printf("     |  Error updating receiver balance.\n");
                sqlite3_finalize(stmt);
                continue;
            }
            sqlite3_finalize(stmt);

            // Insert transaction record
```

```
            char remarks[100] = "TRANSFER";
            sql = "INSERT INTO transactions (from_account_no, to_account_no, amount, remarks) VALUES (?, ?, ?, ?);";
            rc = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
            if (rc != SQLITE_OK) {
                printf("DB error: %s\n", sqlite3_errmsg(db));
                continue;
            }
            sqlite3_bind_int64(stmt, 1, acc_no);
            sqlite3_bind_int64(stmt, 2, recv_acc);
            sqlite3_bind_double(stmt, 3, amount);
            sqlite3_bind_text(stmt, 4, remarks, -1, SQLITE_TRANSIENT);
            if (sqlite3_step(stmt) == SQLITE_DONE) {
                printf("     |  Transaction successful!\n");
            } else {
                printf("     |  Error recording transaction.\n");
            }
            sqlite3_finalize(stmt);

        } else if (cust_menu == 4) {
            logout();
            return;
        } else {
            printf("     |  Invalid choice.\n");
        }
    }
    return;
}

// ===================== Verhoeff Algorithm =====================
//used to verify aadhaar number

// multiplication table
int d[10][10] = {
    {0,1,2,3,4,5,6,7,8,9},
    {1,2,3,4,0,6,7,8,9,5},
    {2,3,4,0,1,7,8,9,5,6},
    {3,4,0,1,2,8,9,5,6,7},
    {4,0,1,2,3,9,5,6,7,8},
    {5,9,8,7,6,0,4,3,2,1},
    {6,5,9,8,7,1,0,4,3,2},
    {7,6,5,9,8,2,1,0,4,3},
    {8,7,6,5,9,3,2,1,0,4},
    {9,8,7,6,5,4,3,2,1,0}
};

// permutation table
int p[8][10] = {
    {0,1,2,3,4,5,6,7,8,9},
    {1,5,7,6,2,8,3,0,9,4},
```

```
    {5,8,0,3,7,9,6,1,4,2},
    {8,9,1,6,0,4,3,5,2,7},
    {9,4,5,3,1,2,6,8,7,0},
    {4,2,8,6,5,7,3,9,0,1},
    {2,7,9,3,8,0,6,4,1,5},
    {7,0,4,6,9,1,3,2,5,8}
};

// inverse table
int inv[10] = {0,4,3,2,1,5,6,7,8,9};
int verhoeff_checksum(const char *num) {
    int c = 0;
    int len = strlen(num);
    for (int i = 0; i < len; i++) {
        int digit = num[len - 1 - i] - '0';
        c = d[c][ p[i % 8][digit] ];
    }
    return c;
}
// Validate Aadhaar (must be 12 digits and pass Verhoeff)
int validate_aadhaar(const char *aadhaar) {
    if (strlen(aadhaar) != 12) {
        return 0; // Aadhaar must be exactly 12 digits
    }
    for (int i = 0; i < 12; i++) {
        if (aadhaar[i] < '0' || aadhaar[i] > '9')
            return 0; // must contain only digits
    }
    return (verhoeff_checksum(aadhaar) == 0);
}


//convert to uppercase
void to_uppercase(char *str) {
    for (int i = 0; str[i]; i++) {
        str[i] = toupper((unsigned char)str[i]);
    }
}

//masking pin
void get_pin(char *pin, int max_len) {
    int i = 0;
    char ch;

    while (i < max_len - 1) {
        ch = getch();           // get a character without showing it

        if (ch == '\r') {       // Enter key pressed
            printf("\n");
            break;
```

```
        } else if (ch == '\b') { // Backspace pressed
            if (i > 0) {
                i--;
                printf("\b \b"); // erase last '*'
            }
        } else if (ch >= '0' && ch <= '9') {
            pin[i++] = ch;
            printf("*");        // print masking character
        }
        // ignore any other characters
    }
    pin[i] = '\0';               // null-terminate
}

// Check if Aadhaar, mobile, or email already exists for a different name
int check_existing_user(const char *name, const char *aadhaar, const char *mobile, const char *email) {
    const char *sql = "SELECT name FROM customers WHERE aadhaar = ? OR mobile = ? OR email = ?;";
    sqlite3_stmt *stmt;
    int rc = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
    if (rc != SQLITE_OK) {
        fprintf(stderr, "SQL error: %s\n", sqlite3_errmsg(db));
        return -1; // DB error
    }

    sqlite3_bind_text(stmt, 1, aadhaar, -1, SQLITE_TRANSIENT);
    sqlite3_bind_text(stmt, 2, mobile, -1, SQLITE_TRANSIENT);
    sqlite3_bind_text(stmt, 3, email, -1, SQLITE_TRANSIENT);

    int allowed = 1;  // assume allowed
    while ((rc = sqlite3_step(stmt)) == SQLITE_ROW) {
        const unsigned char *db_name = sqlite3_column_text(stmt, 0);
        if (strcasecmp((const char*)db_name, name) != 0) {
            // name mismatch → belongs to someone else
            allowed = 0;
            break;
        }
    }

    sqlite3_finalize(stmt);
    return allowed;
}

//==================================CREATE ACCOUNT=======================================
void create_acc(){
    int age,calculated_age,invalid=0;
    int birth_day, birth_month, birth_year;
    char name[50], dob[15], aadhar[20], mobile[12];
    char email[50], address[100], account_type[20];
```

```
553     time_t t = time(NULL);
554     struct tm tm = *localtime(&t);
555     int year = tm.tm_year + 1900;
556 
557 
558     int row_count = 0;
559     const char *sql = "SELECT COUNT(*) FROM customers";
560 
561     rc = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
562     if (rc != SQLITE_OK) {
563         fprintf(stderr, "Failed to prepare statement: %s\n", sqlite3_errmsg(db));
564         sqlite3_close(db);
565         return;
566     }
567 
568     // Execute and get the result
569     if (sqlite3_step(stmt) == SQLITE_ROW) {
570         row_count = sqlite3_column_int(stmt, 0);
571         row_count+=1;
572     }
573 
574     // Show result
575     long long acc_number = (long long)year * 1000000 + row_count;
576     sqlite3_finalize(stmt);
577 
578     printf("      _____________________________________________________________-
___________\n");
579     printf("     |
|\n");
580     printf("     |                        ACCOUNT CREATION PANEL
|\n");
581     printf("     |_____________________________________________________________-
___________|\n");
582     printf("     |
|\n");
583     printf("     |  FULL NAME         : ");      scanf(" %[^\n]", name);
584     printf("     |  DOB (DD/MM/YYYY)  : ");      scanf("%d/%d/%d", &birth_day, &birth_month,
&birth_year);
585     printf("     |  AGE               : ");      scanf("%d", &age);
586     printf("     |  AADHAR NUMBER     : ");      scanf(" %[^\n]", aadhar);
587     printf("     |  MOBILE NUMBER     : +91 ");  scanf(" %[^\n]", mobile);
588     printf("     |  EMAIL             : ");      scanf(" %[^\n]", email);
589     printf("     |  ADDRESS           : ");      scanf(" %[^\n]", address);
590     printf("     |  ACCOUNT TYPE (SAVINGS / CURRENT) : ");     scanf(" %[^\n]",
account_type);
591     printf("     |_____________________________________________________________-
___________|\n\n");
592 
593 
594     //=============verifying DOB & AGE==============
595     // Get current date
```

```
596     t = time(NULL);
597     struct tm today = *localtime(&t);
598 
599     calculated_age = (today.tm_year + 1900) - birth_year;
600     if (birth_month > (today.tm_mon+1) ||
601     (birth_month == (today.tm_mon+1) && birth_day > today.tm_mday)) {
602         calculated_age--;
603     }
604 
605     if (calculated_age != age) {
606         printf("     |  Entered age does NOT match calculated age!
    |\n");
607         return;
608     }
609     if (age < 18) {
610         printf("     |  You must be 18 or older to open an account
    |\n");
611         return;
612     }
613     printf("     |  Age >= 18 verified
    |\n");
614     snprintf(dob, sizeof(dob), "%02d/%02d/%04d", birth_day, birth_month, birth_year);
615 
616     //===========verifying aadhar number==========
617     if (validate_aadhaar(aadhar)) {
618         printf("     |  Aadhar number verified and VALID
    |\n");
619     } else {
620         printf("     |  Aadhar number INVALID
    |\n");
621         invalid=1;
622     }
623 
624     //=============verifying mobile no=============
625     int valid = 1;
626     if (strlen(mobile) != 10) {
627         valid = 0;
628     } else {
629         // Check all characters are digits
630         for (int i = 0; i < 10; i++) {
631             if (!isdigit((unsigned char)mobile[i])) {
632                 valid = 0;
633                 break;
634             }
635         }
636     }
637     switch (valid){
638         case 1: printf("     |  phone number VALID
    |\n");
639                 break;
```

```
640         case 0: printf("     |  phone number INVALID
    |\n");
641                 invalid=2; break;
642         default: break;
643     }
644 
645     //===============verifying email id=================
646     char gmail[] = "@gmail.com";
647     char outlook[] = "@outlook.com";
648 
649     int len = strlen(email);
650     int gmailLen = strlen(gmail);
651     int outlookLen = strlen(outlook);
652 
653     int isValid = 0;
654 
655     // Check for gmail
656     if (len > gmailLen) {
657         int match = 1;
658         for (int i = 0; i < gmailLen; i++) {
659             if (email[len - gmailLen + i] != gmail[i]) {
660                 match = 0;
661                 break;
662             }
663         }
664         if (match == 1) isValid = 1;
665     }
666 
667     // Check for outlook
668     if (len > outlookLen && isValid == 0) {
669         int match = 1;
670         for (int i = 0; i < outlookLen; i++) {
671             if (email[len - outlookLen + i] != outlook[i]) {
672                 match = 0;
673                 break;
674             }
675         }
676         if (match == 1) isValid = 1;
677     }
678 
679     if (isValid == 1){
680         printf("     |  Email id verified and VALID
    |\n");
681     }else{
682         printf("     |  Email id INVALID
    |\n");
683         invalid=3;
684     }
685 
686     while (invalid != 0) {
687         printf("     |  INVALID data entered. Please try again.\n");
```

```c
        create_acc();
        return;
    }

    //Fake account check
    if (!check_existing_user(name, aadhar, mobile, email)) {
        printf("\nAccount creation failed: Aadhaar/Mobile/Email already linked to another person.\n");
        create_acc();
        return;  // stop creation}
    }

    //ENTER PIN
    char pin[10],con_pin[10];
    while (1) {
        printf("     |
|\n");
        printf("     |  ENTER PIN          : ");    get_pin(pin, sizeof(pin));
        printf("     |  CONFIRM PIN        : ");    get_pin(con_pin, sizeof(con_pin));
        if (strlen(pin)!=6 || strlen(con_pin)!=6) {
            printf("     |  PIN must be exactly 6 digits long
|\n");
            printf("     |  Please try entering the PIN again
|\n");
            continue;
        }

        // Check match
        if (strcmp(pin,con_pin)!=0){
            printf("     |  PIN and Confirm PIN do not match
|\n");
            printf("     |  Please try entering the PIN again
|\n");
            continue;}
        printf("     |  PIN set successfully!
|\n");
        break;
    }

    int pin_test=atoi(pin);
    int encrypt=pin_test<<3;
    char encrypted_pin[10];
    sprintf(encrypted_pin, "%d", encrypt);
    to_uppercase(name);
    to_uppercase(address);
    to_uppercase(account_type);

    const char *insert_sql = "INSERT INTO customers "
                      "(account_no, name, dob, aadhaar, mobile, email, address, account_type, pin) "
                      "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
```

```
731
732     rc = sqlite3_prepare_v2(db, insert_sql, -1, &stmt, NULL);
733     if (rc != SQLITE_OK) {
734         fprintf(stderr, "SQL error: %s\n", sqlite3_errmsg(db));
735         sqlite3_close(db);
736         return;
737     }
738
739     sqlite3_bind_int64(stmt, 1, acc_number);
740     sqlite3_bind_text(stmt, 2, name, -1, SQLITE_TRANSIENT);
741     sqlite3_bind_text(stmt, 3, dob, -1, SQLITE_TRANSIENT);
742     sqlite3_bind_text(stmt, 4, aadhar, -1, SQLITE_TRANSIENT);
743     sqlite3_bind_text(stmt, 5, mobile, -1, SQLITE_TRANSIENT);
744     sqlite3_bind_text(stmt, 6, email, -1, SQLITE_TRANSIENT);
745     sqlite3_bind_text(stmt, 7, address, -1, SQLITE_TRANSIENT);
746     sqlite3_bind_text(stmt, 8, account_type, -1, SQLITE_TRANSIENT);
747     sqlite3_bind_text(stmt, 9, encrypted_pin, -1, SQLITE_TRANSIENT);
748
749     if (sqlite3_step(stmt) != SQLITE_DONE) {
750         fprintf(stderr, "Insert failed: %s\n", sqlite3_errmsg(db));
751     } else {
752         printf("     |_____________________________________________________-
___________|\n\n");
753         printf("     |  Account Created Successfully!
|\n");
754     }
755
756     sqlite3_finalize(stmt);
757     view_details(name);
758     int option=admin_option();
759     switch (option) {
760         case 1: create_acc(); break;
761         case 2: customer_details(); break;
762         case 3: delete_acc(); break;
763         case 4: logout(); break;
764         default: break;
765     }
766 }
767
768 void customer_details(){
769     char search[50];
770     printf("      _____________________________________________________-
___________\n");
771     printf("     |
|\n");
772     printf("     |                        CUSTOMER DETAILS PANEL
|\n");
773     printf("     |_____________________________________________________-
___________|\n");
774     printf("     |
|\n");
```

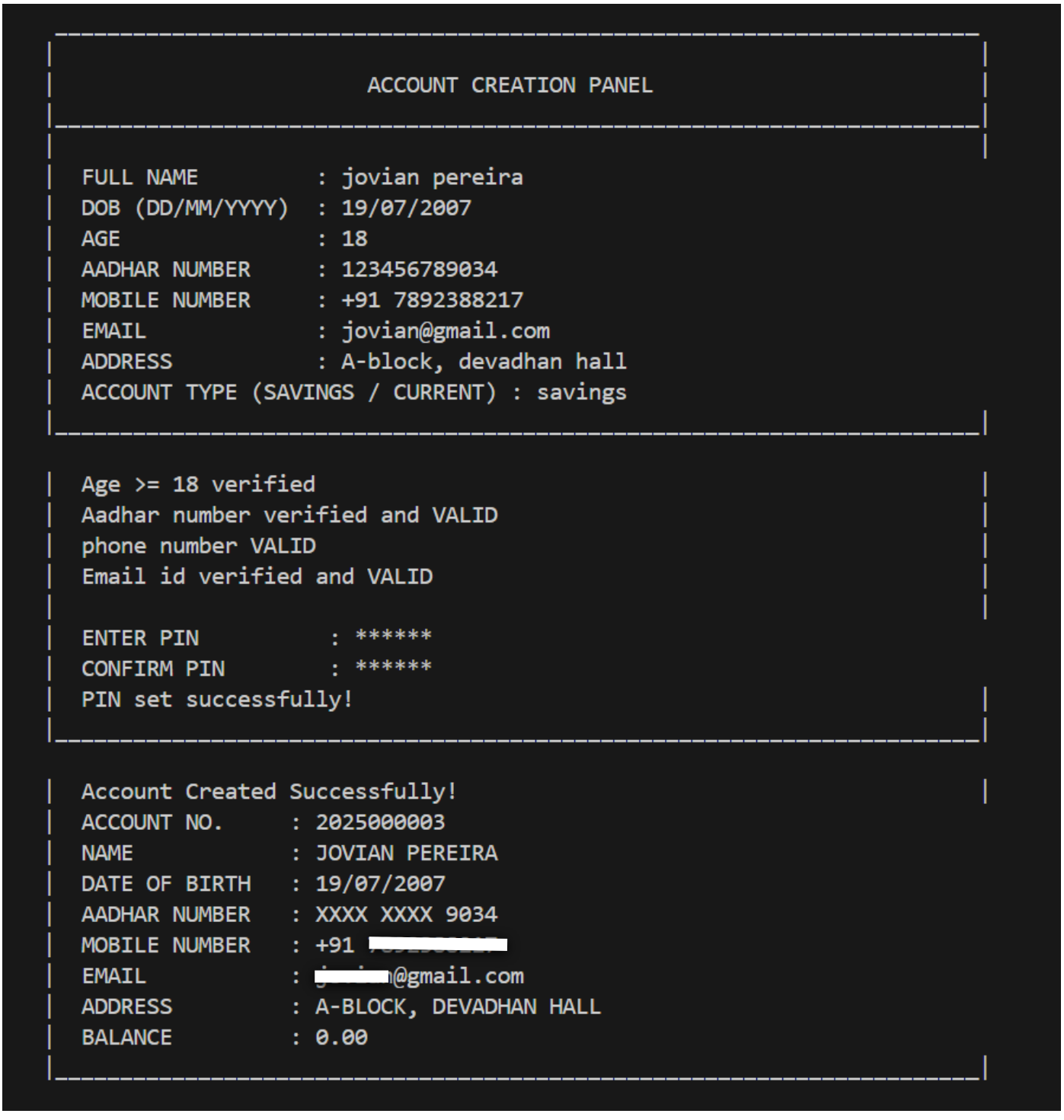
```
775     printf("     |  ENTER CUSTOMER NAME : ");     scanf(" %[^\n]", search);
776     printf("     |_____________________________________________________________-
___________|\n");
777     view_details(search);
778     int option=admin_option();
779     switch (option) {
780         case 1: create_acc(); break;
781         case 2: customer_details(); break;
782         case 3: delete_acc(); break;
783         case 4: logout(); break;
784         default: break;
785     }
786 }
787 
788 void delete_acc(){
789     int acc_no;
790     char pin[10];
791 
792     printf("      _____________________________________________________________-
___________\n");
793     printf("     |
|\n");
794     printf("     |                        ACCOUNT DELETION PANEL
|\n");
795     printf("     |_____________________________________________________________-
___________|\n");
796     printf("     |       ACCOUNT DELETION TO BE DONE IN PRESENCE OF ACCOUNT HOLDER
|\n");
797     printf("     |
|\n");
798     printf("     |  ENTER ACCOUNT NUMBER : ");     scanf("%d",&acc_no);
799     printf("     |  ENTER PIN            : ");     get_pin(pin, sizeof(pin));
800     printf("     |_____________________________________________________________-
___________|\n");
801 
802 
803     // Step 1: Fetch encrypted PIN from database
804     const char *sql = "SELECT pin FROM customers WHERE account_no = ?;";
805     int rc = sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);
806     if (rc != SQLITE_OK) {
807         fprintf(stderr, "Failed to prepare statement: %s\n", sqlite3_errmsg(db));
808         return;
809     }
810 
811     sqlite3_bind_int(stmt, 1, acc_no);
812 
813     rc = sqlite3_step(stmt);
814     if (rc == SQLITE_ROW) {
815         const unsigned char *db_pin_text = sqlite3_column_text(stmt, 0);
816         int stored_enc = atoi((const char*)db_pin_text);
817         int stored_pin = stored_enc >> 3; // decrypt
```

```
818
819         sqlite3_finalize(stmt);
820
821         int entered_int = atoi(pin);
822         if (entered_int == stored_pin) {
823             // Step 2: PIN verified → delete account
824             const char *del_sql = "DELETE FROM customers WHERE account_no = ?;";
825             rc = sqlite3_prepare_v2(db, del_sql, -1, &stmt, NULL);
826             if (rc != SQLITE_OK) {
827                 fprintf(stderr, "Failed to prepare delete statement: %s\n",
    sqlite3_errmsg(db));
828                 return;
829             }
830
831             sqlite3_bind_int(stmt, 1, acc_no);
832
833             if (sqlite3_step(stmt) == SQLITE_DONE) {
834                 printf("     |   ACCOUNT DELETED SUCCESSFULLY
    |\n");
835             } else {
836                 printf("     |   Failed to delete account.
    |\n");
837             }
838
839             sqlite3_finalize(stmt);
840         } else {
841             printf("     |   Incorrect PIN. Account deletion aborted.
    |\n");
842         }
843     } else {
844         printf("     |   No account found with number %d.              |\n", acc_no);
845         sqlite3_finalize(stmt);
846     }
847     int option=admin_option();
848     switch (option) {
849         case 1: create_acc(); break;
850         case 2: customer_details(); break;
851         case 3: delete_acc(); break;
852         case 4: logout(); break;
853         default: break;
854     }
855 }
856
857 void logout(){
858     choice();
859     return;
860 }
861
862 int main() {
863     // Open database named bankdb.db
864     const char* filePath = "D:\\Ryan\\projects\\BankMate\\bankdb.db";
```

```
865     int rc = sqlite3_open(filePath, &db);
866
867     if (rc != SQLITE_OK) {
868         fprintf(stderr, "Cannot open database: %s\n", sqlite3_errmsg(db));
869         sqlite3_close(db);
870         return 1;
871     }
872     printf("Connection successful\n");
873
874     while (1) {
875         int option = choice();
876         switch (option) {
877             case 1: admin(); break;
878             case 2: customers(); break;
879             case 3: sqlite3_close(db); return 0;
880             default: break;
881         }
882     }
883     return 0;
884 }
885
```

Sample Output

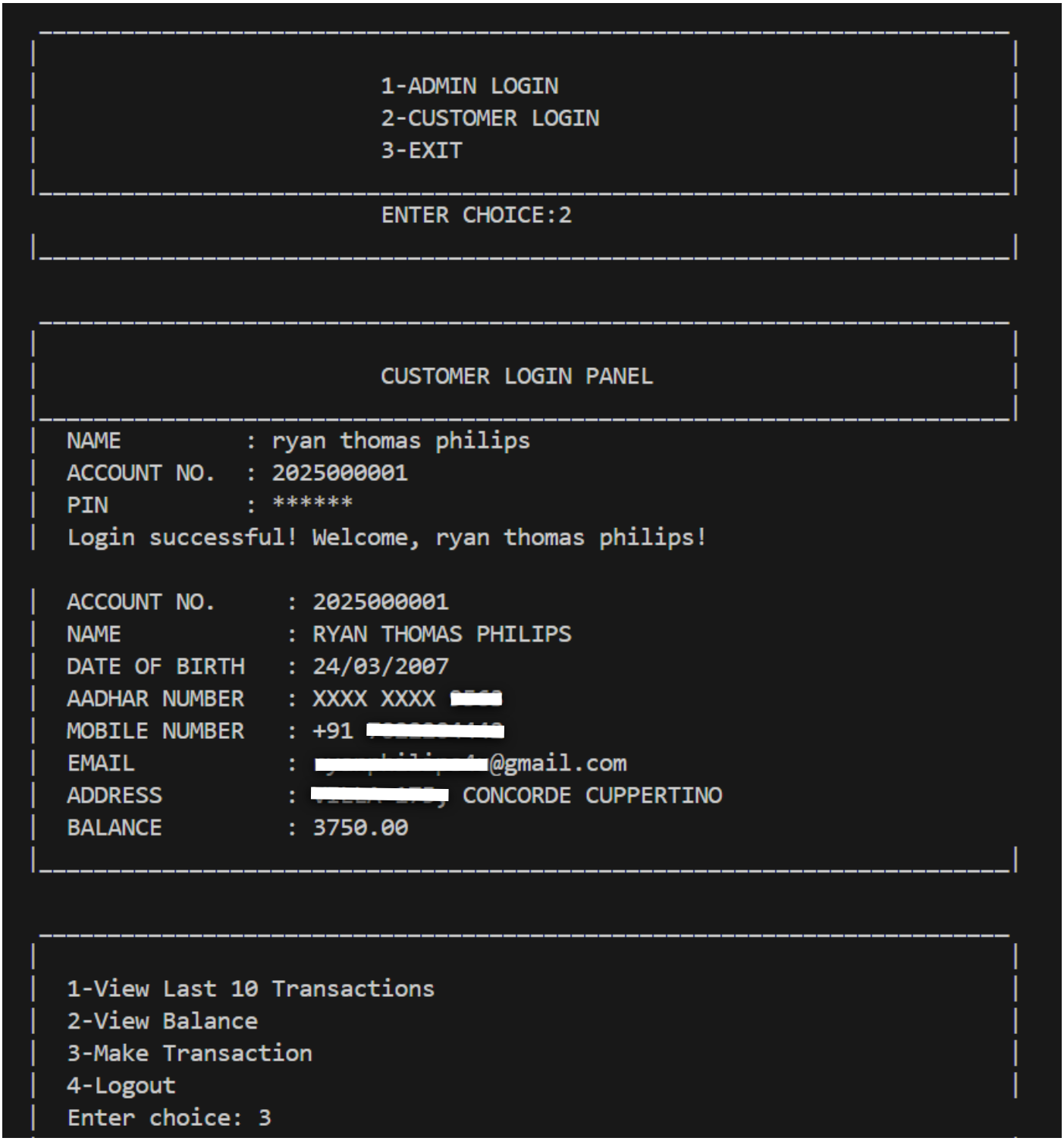
```
PS D:\Ryan\projects\BankMate\output> & .\'bankmate.exe'
Connection successful
 ____________________________________________________________________________
|                                                                            |
|                          1-ADMIN LOGIN                                     |
|                          2-CUSTOMER LOGIN                                  |
|                          3-EXIT                                            |
|____________________________________________________________________________|
                           ENTER CHOICE:1
 ____________________________________________________________________________
|                                                                            |
|                          ADMIN LOGIN PANEL                                 |
|____________________________________________________________________________|
|                                                                            |
|  USERNAME : ryan                                                           |
|  PASSWORD : admin123                                                       |
|____________________________________________________________________________|
|                                                                            |
|  Login Successful!                                                         |
|  Admin ID   : 1024                                                         |
|  Welcome, ryan!                                                            |
|                                                                            |
|  1-CREATE BANK ACCOUNT                                                     |
|  2-CUSTOMER ACCOUNT DETAILS                                                |
|  3-DELETE BANK ACCOUNT                                                     |
|  4-LOG OUT                                                                 |
|____________________________________________________________________________|
|       ENTER CHOICE:1                                                       |
|____________________________________________________________________________|
```


ACCOUNT CREATION PANEL
FULL NAME : jovian pereira
DOB (DD/MM/YYYY) : 19/07/2007
AGE : 18
AADHAR NUMBER : 123456789034
MOBILE NUMBER : +91 7892388217
EMAIL : jovian@gmail.com
ADDRESS : A-block, devadhan hall
ACCOUNT TYPE (SAVINGS / CURRENT) : savings
Age >= 18 verified
Aadhar number verified and VALID
phone number VALID
Email id verified and VALID
ENTER PIN : ******
CONFIRM PIN : ******
PIN set successfully!
Account Created Successfully!
ACCOUNT NO. : 2025000003
NAME : JOVIAN PEREIRA
DATE OF BIRTH : 19/07/2007
AADHAR NUMBER : XXXX XXXX 9034
MOBILE NUMBER : +91
EMAIL : @gmail.com
ADDRESS : A-BLOCK, DEVADHAN HALL
BALANCE : 0.00

```
|  1-CREATE BANK ACCOUNT                                                    |
|  2-CUSTOMER ACCOUNT DETAILS                                               |
|  3-DELETE BANK ACCOUNT                                                    |
|  4-LOG OUT                                                                |
|___________________________________________________________________________|
|      ENTER CHOICE:2                                                       |
|___________________________________________________________________________|


 ___________________________________________________________________________
|                                                                           |
|                        CUSTOMER DETAILS PANEL                             |
|___________________________________________________________________________|
|                                                                           |
|  ENTER CUSTOMER NAME : hrishikesh.p.s                                     |
|___________________________________________________________________________|
|  ACCOUNT NO.      : 2025000002                                            |
|  NAME             : HRISHIKESH.P.S                                        |
|  DATE OF BIRTH    : 13/09/2007                                            |
|  AADHAR NUMBER    : XXXX XXXX [redacted]                                  |
|  MOBILE NUMBER    : +91 [redacted]                                        |
|  EMAIL            : [redacted]h@gmail.com                                 |
|  ADDRESS          : DEVADHAN HALL                                         |
|  BALANCE          : 1250.00                                               |
|___________________________________________________________________________|


|  1-CREATE BANK ACCOUNT                                                    |
|  2-CUSTOMER ACCOUNT DETAILS                                               |
|  3-DELETE BANK ACCOUNT                                                    |
|  4-LOG OUT                                                                |
|___________________________________________________________________________|
|      ENTER CHOICE:4                                                       |
|___________________________________________________________________________|
```


1-ADMIN LOGIN
2-CUSTOMER LOGIN
3-EXIT
ENTER CHOICE:2
CUSTOMER LOGIN PANEL
NAME : ryan thomas philips
ACCOUNT NO. : 2025000001
PIN : ******
Login successful! Welcome, ryan thomas philips!
ACCOUNT NO. : 2025000001
NAME : RYAN THOMAS PHILIPS
DATE OF BIRTH : 24/03/2007
AADHAR NUMBER : XXXX XXXX
MOBILE NUMBER : +91
EMAIL : @gmail.com
ADDRESS : CONCORDE CUPPERTINO
BALANCE : 3750.00
1-View Last 10 Transactions
2-View Balance
3-Make Transaction
4-Logout
Enter choice: 3

```
 ____________________________________________________________________________
|                                                                            |
|  1-View Last 10 Transactions                                               |
|  2-View Balance                                                            |
|  3-Make Transaction                                                        |
|  4-Logout                                                                  |
|  Enter choice: 3                                                           |
|____________________________________________________________________________|
|  Enter receiver's name: jovian pereira
|  Enter last 4 digits of receiver's account: 0003
|  Enter amount to send: 500
|  Enter your PIN to confirm: ******

|  Transferring 500.00 to jovian pereira (Acc: 2025000003)...
|  Transaction successful!

 ____________________________________________________________________________
|                                                                            |
|  1-View Last 10 Transactions                                               |
|  2-View Balance                                                            |
|  3-Make Transaction                                                        |
|  4-Logout                                                                  |
|  Enter choice: 2                                                           |
|____________________________________________________________________________|

|  Current Balance: 3250.00
```

```
 ____________________________________________________________________
|                                                                    |
|  1-View Last 10 Transactions                                       |
|  2-View Balance                                                    |
|  3-Make Transaction                                                |
|  4-Logout                                                          |
|  Enter choice: 1                                                   |
|____________________________________________________________________|

 ____________________________________________________________________________________________________________
|  Date/Time           |  From                |  To                  |  Amount   |  Remarks             |
|----------------------|----------------------|----------------------|-----------|----------------------|
| 2025-09-17 21:24:28  | RYAN THOMAS PHILIPS  | JOVIAN PEREIRA       | 500.00    | TRANSFER
| 2025-09-17 18:59:34  | RYAN THOMAS PHILIPS  | HRISHIKESH.P.S       | 200.00    | TRANSFER
| 2025-09-17 18:36:37  | RYAN THOMAS PHILIPS  | HRISHIKESH.P.S       | 1000.00   | TRANSFER
| 2025-09-17 13:14:39  | HRISHIKESH.P.S       | RYAN THOMAS PHILIPS  | 350.00    | TRANSFER
| 2025-09-17 13:09:55  | RYAN THOMAS PHILIPS  | HRISHIKESH.P.S       | 200.00    | TRANSFER
|____________________________________________________________________________________________________________|


 ____________________________________________________________________
|                                                                    |
|  1-View Last 10 Transactions                                       |
|  2-View Balance                                                    |
|  3-Make Transaction                                                |
|  4-Logout                                                          |
|  Enter choice: 4                                                   |
|____________________________________________________________________|

 ____________________________________________________________________
|                                                                    |
|                        1-ADMIN LOGIN                               |
|                        2-CUSTOMER LOGIN                            |
|                        3-EXIT                                      |
|____________________________________________________________________|
                         ENTER CHOICE:3
|____________________________________________________________________|
```

CONCLUSION

This banking project successfully demonstrates the essential functionalities of a digital banking system, including secure customer account creation, authentication, transaction handling, and data storage using a relational database. It highlights the practical application of programming logic, structured system design, and secure data handling through the use of PIN encryption implemented with bitwise operations. These components together form a reliable and efficient foundation for managing fundamental banking operations.

Looking ahead, the system can be further enhanced with additional features to improve functionality and user experience. A graphical user interface can be developed for more intuitive interaction, and modules for transaction history, mini-statements, and account summaries can be incorporated. Basic notification systems such as SMS or email alerts using available APIs can be added to keep users informed of account activity. Hosting the application on a local or cloud-based server to support multi-user access can also be explored. These enhancements would help transform the project into a more comprehensive and user-friendly banking solution.