Index

Acknowledgement

Apart from the efforts of me, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I would like to express a deep sense of thanks & gratitude to my project guide Aiysha Siddiqui for guiding me immensely through the course of this project whose constructive advice & constant motivation have been responsible for the successful completion of this project.

My sincere thanks go to Sharatdeep Mathur, our Principal and Sugandha Khopkar, our Vice Principal for their co-ordination in extending every possible support for the completion of the project.

I express my heartfelt gratitude to my parents for constant encouragement while carrying out this project.

Last but not the least; I would like to thank all those who have helped directly or indirectly towards the completion of the project.

Jaydeep Sanjay Solanki

CLASS: XII A

System Requirements

HARDWARE:

1. PROCESSOR: INTEL®CORE™ I3-2100 CPU@3.10GHz
2. RAM:3GB
3. SYSTEM ARCHITECTURE:64 BIT OPERATING SYSTEM
4. KEYBOARD
5. MOUSE

SOFTWARE:

1. OPERATING SYSTEM: ANY OPERATING SYSTEM(x64 or x86) WITH PYTHON (VERSION>3.2.0)
2. MYSQL(OR ANY OTHER SOFTWARE LIKE WAMP SUPPORTING MYSQL)
3. MYSQL PHYTHON CONNECTOR IS COMPULSORY NEEDED.

Objective Of Project

This project is developed in Python platform with MySQL database as backend and has been developed to clone the process of banking system

It has two users : Customer and Admin. It is included with integrated email verification system using a module.

Proposed System

All the modules of systems have been automated and efforts have been made to minimize the manual working.

MODULES:

i USER:

- Login: To start using this system user has to create an account with his/her details .This data will be stored in the database.
- Withdraw: The respective costumer can enter the details of his/her account and withdraw the amount entered from his/her account.
- Deposit: The respective costumer can provide his/her details and deposit money in his/her account retailer.
- Transfer: The respective costumer can provide his/her details along with the user to which the customer wants to transfer money.

ii Admin:

- LOGIN: Admin can login with his/her account.
- Logs: Check the logs like who registered and who deposited how much money . Note the admin don't have permission to see the user password.
    i. UPDATED
    ii. DEPOSITED
    iii. WITHDRAWN
    iv. TRANSFERRED
    v. RECEIVED
    vi. REGISTERED
    vii. DELETED
    viii. LOGIN
    ix. ALL
- Delete: The admin can delete the user account.

Benefits Of Proposed System

1) **Less Paper Work:** The paper work is reduced to minimal level. Computer prepares the lists of food items.

2) **No Manual Work:** There is no manual work. All the processes are done through computer.

3) **Record of customers:** There is record of all the customers who got registered with their addresses and mobile numbers.

4) **Review Maintenance is Easier:** Review can now easily be maintained by producing a report with a format of adding customers records.

5) **Data Is Not Scattered:** Data is now stored at one place. Any information regarding anything can be easily available to the user.

6) **Flexibility:** The system is more flexible than the manual system being used presently.

7) **Beneficial:** The system is easy to use and reduces the user's workload a lot. It provides timely and accurate information and there is automatic generation of orders.

8) **Greater reach:** This system provides a wide reach of retailer to a lot of customers who can simply order food without any problems with the help of this simple proposed system.

Description & Conclusion

This project was developed from an idea of cloning the bank management system which I saw in YouTube. Adding email verification was my idea because as in real life the user has to verify it by phone number but doing phone verification system is high level coding so I chose using email verification system.

At the end I would like to conclude that I loved coding this system. Sometimes I went in trouble but I got helped from my mam and youtube.

Some Main Functions from Source Code

1. Most Important and Basic Function for Running SQL Queries through python

```
def sql_query(query, function, password="", dbname="bank"):
    db = mysql.connector.connect(
        host="localhost", user="root", password=password, database=dbname)
    try:
        cursor = db.cursor()
        if function == "execute":
            cursor.execute(query)
            db.commit()
            db.close()
        elif function == "extract":
            cursor.execute(query)
            return cursor.fetchall()
    except Exception as error:

        clear()
        print(error)
        print("Error in sql_query")
        exc_type, exc_obj, exc_tb = sys.exc_info()
        file_name = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
        print(f"""
Exception Type:{exc_type}
File Name:{file_name}
Error Occurred Line Number: {exc_tb.tb_lineno}""")
```

2. Function For Logging all the task performed

```
def log_the_task(USER_BANK_ID, USER_ACCOUNT_ID, NAME, EMAIL, PREV_AMOUNT, NEW_BAL, TASK_PERFORMED,
                 TIMESTAMP=datetime.now().strftime('%I:%M:%S %p  %Y/%m/%d')):
    query_execute = f"INSERT INTO LOGS (USER_BANK_ID,USER_ACCOUNT_ID,NAME,EMAIL,PREV_AMOUNT,NEW_BAL,TASK_PERFORMED,TIMESTAMP)
VALUES('{USER_BANK_ID}','{USER_ACCOUNT_ID}','{NAME}','{EMAIL}','{PREV_AMOUNT}','{NEW_BAL}','{TASK_PERFORMED}','{TIMESTAMP}')"
    sql_query(query_execute, "execute")
```

3. Email Related Functions

```
def sending_otp_to_user(user_email_id_to_send, Function, FUNCTION_TO_DO_IF_CORRECT=False):
    call_the_name("OTP VERIFICATION PROCEDURE")
    send_otp = str(email_verification(user_email_id_to_send, "User"))
    enter_otp = input(f"Please enter otp sent to {user_email_id_to_send}  : ")
    while enter_otp != send_otp:

        print(
            f"The otp entered is not same to which we sent to {user_email_id_to_send}")
        print(
            """Do you want us to send the otp again to
               1. similar email
               2. you want to change email""")
        your_choice = input("Choice [1-2] : ")
        if your_choice == "1":
            call_the_name("OTP VERIFICATION PROCEDURE")
            send_otp = str(email_verification(user_email_id_to_send, "User"))
            enter_otp = input(
                f"Please enter otp sent to {user_email_id_to_send} : ")
        else:
            clear()
            user_register_protocol()

    if enter_otp == send_otp:
        if FUNCTION_TO_DO_IF_CORRECT is not False:
            clear()
            print("Email successfully verified you can now proceed ahead.")
            sleep(5)
            Function(user_email_id_to_send)
        else:
            clear()
            print("Email successfully verified you can now proceed ahead.")
            sleep(5)
            Function()
```

```
def email_verification(receiver_email, receiver_name):
    try:
        sender_email = secrets.email()
        sender_email_password = secrets.email_password()
        def otp():
            main_otp = ""
            for _ in range(4):
                otp_1 = random.choices([1, 2, 3, 4, 5, 6, 7, 8, 9, 0])
                main_otp += str(otp_1[0])
            return main_otp

        otp_is = otp()
        msg = message_content_send(
            sender_email, receiver_name, receiver_email, otp_is)

        with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:

            smtp.login(sender_email, sender_email_password)
            smtp.send_message(msg)
        return otp_is
    except:
        clear()
        print("Error in email_verification")
        exc_type, exc_obj, exc_tb = sys.exc_info()
        file_name = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
        print(f"""
Exception Type:{exc_type}
File Name:{file_name}
Error Occurred Line Number: {exc_tb.tb_lineno}""")

        sleep(5)
        raise_error(interface_director)
```

4. Function to direct user

```
def user_main_interface():
    clear()
    try:
        call_the_name("user interface")
        print(
            """
        1. Login
        2. Register
        3. Delete
        4. Update
        5. Quit""")
        print("_" * 30)
        return input("""Choice [1 or 2 or 3 or 4 or 5] : """)
    except:
        clear()
        print("Error in user_main_interface")
        exc_type, exc_obj, exc_tb = sys.exc_info()
        file_name = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
        print(f"""
Exception Type:{exc_type}
File Name:{file_name}
Error Occurred Line Number: {exc_tb.tb_lineno}""")

        sleep(5)
        raise_error(user_main_interface_director)
```

```
    def user_main_interface_director():
        clear()
        try:
            user_inter = user_main_interface()
            if user_inter == "1":
                user_login_protocol()
            elif user_inter == "2":
                user_register_protocol()
            elif user_inter == "3":
                user_delete_protocol()
            elif user_inter == "4":
                user_update_protocol()
            elif user_inter == "5":
                clear()
                interface_director()
        except:
            clear()
            print("Error in user_main_interface_director")
            exc_type, exc_obj, exc_tb = sys.exc_info()
            file_name = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
            print(f"""
    Exception Type:{exc_type}
    File Name:{file_name}
    Error Occurred Line Number: {exc_tb.tb_lineno}""")

            sleep(5)
            raise_error(user_main_interface_director)
```

5. Function for logging in valid user

```
def user_login_protocol():
    clear()
    try:
        call_the_name("LOGIN PAGE")
        user_email = input("Please type your registered email address: ")
        user_email_fetch_query = "SELECT EMAIL_ID FROM USERS"
        user_email_fetch = sql_query(user_email_fetch_query, "extract")
        user_email_fetch_list = [i[0] for i in user_email_fetch]

        if user_email not in user_email_fetch_list:
            print(
                "Seems email is not registered you can not login without registering .")
            sleep(5)
            user_main_interface_director()
        else:
            user_id = int(input("Please enter user id : "))
            user_password = int(input("Please type your password [must be number] : "))
            user_auth_query = f"SELECT EMAIL_ID,PASSWORD,NAME FROM USERS WHERE ID={user_id}"
            user_auth_execute = sql_query(user_auth_query, "extract")
            if user_auth_execute[0][0] == user_email and user_auth_execute[0][1] == user_password:
                clear()
                print(f"Successfully logged in as {user_auth_execute[0][2]}")
                sleep(2)
                clear()
                print(f"Welcome {user_auth_execute[0][2]}")
                print("Redirecting to your user choices page .")
                log_the_task(user_id, 0, user_auth_execute[0][2], user_email, 0, 0, f"USER LOGIN SUCCESSFULLY")
                sleep(5)
                choices_main_interface_director()
            else:
                print("Invalid Credentials")
                user_login_protocol()
    except:
        clear()
        print("Error in user_login_protocol")
        exc_type, exc_obj, exc_tb = sys.exc_info()
        file_name = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
        print(f"""
Exception Type:{exc_type}
File Name:{file_name}
Error Occurred Line Number: {exc_tb.tb_lineno}""")

        sleep(5)
        raise_error(user_login_protocol)
```

6. Admin Interface Director

```
def admin_main_interface():
    clear()
    try:
        call_the_name("ADMIN INTERFACE PAGE")
        print(
            """
        1. Login
        2. Register
        3. Delete
        4. Quit""")
        print("_" * 30)
        return input("""Choice [1 or 2 or 3 or 4] : """)
    except:
        clear()
        print("Error in admin_main_interface")
        exc_type, exc_obj, exc_tb = sys.exc_info()
        file_name = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
        print(f"""
Exception Type:{exc_type}
File Name:{file_name}
Error Occurred Line Number: {exc_tb.tb_lineno}""")

        sleep(5)
        raise_error(admin_main_interface_director)


def admin_main_interface_director():
    clear()
    try:
        admin_inter = admin_main_interface()
        print(admin_inter)
        if admin_inter == "1":
            admin_login()
        elif admin_inter == "2":
            admin_register()
        elif admin_inter == "3":
            admin_delete()
        elif admin_inter == "4":
            clear()
            interface_director()
    except:
        clear()
        print("Error in admin_main_interface_director")
        exc_type, exc_obj, exc_tb = sys.exc_info()
        file_name = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
        print(f"""
Exception Type:{exc_type}
File Name:{file_name}
Error Occurred Line Number: {exc_tb.tb_lineno}""")

        sleep(5)
        raise_error(admin_main_interface_director)
```

7. Logging in valid registered admin

```
def admin_login():
    clear()
    try:
        call_the_name("ADMIN LOGIN PAGE")
        admin_id = int(input("Please enter admin id : "))
        admin_email = input("Please type your registered email address: ")
        admin_password = int(input("Please type your password[must be number] : "))
        auth_query = f"SELECT EMAIL,PASSWORD,NAME FROM ADMINS WHERE ID={admin_id}"
        auth_execute = sql_query(auth_query, "extract")
        if auth_execute[0][0] == admin_email and auth_execute[0][1] == admin_password:
            clear()
            print(f"Successfully logged in as {auth_execute[0][2]}")
            sleep(2)
            print(f"Welcome {auth_execute[0][2]}")
            print("Redirecting to your admin choices page .")
            sleep(5)
            log_the_task(admin_id, 0, f"ADMIN {auth_execute[0][2]}", admin_email, 0, 0, f"ADMIN LOGIN IN SUCCESSFULLY")
            timewise_report()
        else:
            print("Invalid Credentials")
            admin_login()
    except:
        clear()
        print("Error in admin_login")
        exc_type, exc_obj, exc_tb = sys.exc_info()
        file_name = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
        print(f"""
Exception Type:{exc_type}
File Name:{file_name}
Error Occurred Line Number: {exc_tb.tb_lineno}""")

        sleep(5)
        raise_error(admin_login)
```

8. Functions Available for both user and admin are

   i. Admin Choices

```
def admin_choicer():
    clear()
    try:
        call_the_name("ADMIN CHOICES PAGE")
        print(
            """
        1. Daily Report
        2. Monthly Report
        3. Yearly Report
        4. Money Deposit Report
        5. Money Withdraw Report
        6. Money Transfer Report
        7. Money Received Report
        8. Registered Accounts Information
        9. Login Accounts Information
        10. Deleted Accounts Information
        11. All Logs
        12. Specify Report
        13. Details Updated
        """)
        return input("\tEnter your choice 1 to 13 : ")
    except:
        clear()
        print("Error in admin_choicer")
        exc_type, exc_obj, exc_tb = sys.exc_info()
        file_name = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
        print(f"""
Exception Type:{exc_type}
File Name:{file_name}
Error Occurred Line Number: {exc_tb.tb_lineno}""")

        sleep(5)
        raise_error(timewise_report)
```

ii. User Choices

```
def choices_main_interface():
    clear()
    try:
        call_the_name("USER ACCOUNT INTERFACE")
        print(
            """
        1. Deposit
        2. Withdraw
        3. Transfer
        4. Quit
        """)
        print("_" * 30)
        return input("""Choice [1 or 2 or 3 or 4 or 5] : """)

    except:
        clear()
        print("Error in choices_main_interface")
        exc_type, exc_obj, exc_tb = sys.exc_info()
        file_name = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
        print(f"""
Exception Type:{exc_type}
File Name:{file_name}
Error Occurred Line Number: {exc_tb.tb_lineno}""")

        sleep(5)
        raise_error(choices_main_interface_director)

def choices_main_interface_director():
    clear()
    try:
        user_acc_inter = choices_main_interface()
        if user_acc_inter == "1":
            choice_deposit()
        elif user_acc_inter == "2":
            choice_withdraw_protocol()
        elif user_acc_inter == "3":
            choice_transfer_protocol()
        elif user_acc_inter == "4":
            clear()
            account_main_interface_director()
    except:
        clear()
        print("Error in choices_main_interface_director")
        exc_type, exc_obj, exc_tb = sys.exc_info()
        file_name = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
        print(f"""
Exception Type:{exc_type}
File Name:{file_name}
Error Occurred Line Number: {exc_tb.tb_lineno}""")

        sleep(5)
        raise_error(choices_main_interface_director)
```

9. Major imports

```
try:
    import get_pip
    import os
    import random
    import secrets
    import smtplib
    import subprocess
    import sys
    from datetime import date, datetime
    from email.message import EmailMessage
    from time import sleep
except Exception as Error:
    print(Error)


def package_installer(package_name):
    return subprocess.call([sys.executable, "-m", "pip", "install", package_name])


try:
    import mysql.connector
    from tabulate import tabulate
except Exception as e:
    print(e)
    try:
        import pip

        package_installer("mysql-connector-python")
        package_installer("tabulate")
    except Exception as e:
        print(e)
        get_pip.main()
        try:
            import pip

            package_installer("mysql-connector-python")
            package_installer("tabulate")
        except Exception as e:
            print(e)
```

Used for installing external packages used in program

Output Of Code

```
================================================================================
            W E L C O M E   T O   U R M I   B A N K I N G
================================================================================
You will find a file named 'secrets.py'
Open it and update that file to the credentials,Instructions and helpful links are shared in secrets.py  ...



Press enter to move ahead...
```

```
================================================================================================================
            I N V A L I D   E M A I L   O R   P A S S   I N   ' S E C R E T S . P Y '
================================================================================================================
Please enter the valid email or password in 'secrets.py' for proper functioning !
After entering details re run this program!
```

It checks for credentials if it isn't edited it throws error and closes the program

```
================================================================================
            T A B L E   C R E A T I O N   P R O C E D U R E
================================================================================
            Creating database..
            Creating tables ..
            Both Database Tables have been created..

                1. Database :
                    > BANK
                2. Tables  :
                    > USERS
                    > ACCOUNT_DETAILS
                    > ADMINS
                    > LOGS
```

Automatic Table Creation

```
==============================
          L O G I N
==============================

    1. User Interface
    2. Admin Interface

_____________________________
Choice [1 or 2] :
```

1st page shown to the user who ran this program

```
================================================================
            U S E R   I N T E R F A C E
================================================================

        1. Login
        2. Register
        3. Delete
        4. Update
        5. Quit

_____________________________
Choice [1 or 2 or 3 or 4 or 5] : 2
```

Directed to this page if user choose 1 in login page

```
================================================================================
                REGISTRATION AUTH PROTOCOL
================================================================================
We will send otp to the email you provide below so make sure you enter valid email id.
Please type your email address:
```

Foremost condition for registering new user.

Your otp for Urmi Bank is 1602 - OTP VERIFICATION Hi, User Your otp for email :

```
================================================================================
                OTP VERIFICATION PROCEDURE
================================================================================
Please enter otp sent to [redacted]@gmail.com  : 1602

Email successfully verified you can now proceed ahead.


=========================================================================
                REGISTRATION PAGE
=========================================================================
Please enter your name : Jaydeep
Please enter your Date of birth: 10-05-2004
Please type your password[must be number] : 9898
Please type your phone number: 1234567891
Please enter your aadhaar number: 1234-1234-1234
Please enter your gender: male
Successfully registered as Jaydeep having email [redacted]@gmail.com and password 9898
Your user id is 1
Remember this user id it will be used in the user confirmation.
Redirecting to your user choices page.

================================================================================
                USER ACCOUNT INTERFACE
================================================================================

        1. Deposit
        2. Withdraw
        3. Transfer
        4. Quit


Choice [1 or 2 or 3 or 4 or 5] : 4

================================================================================
                USER ACCOUNT INTERFACE
================================================================================

        1. Login
        2. Register
        3. Delete
        4. Quit

Choice [1 or 2 or 3 or 4] :
```

Similarly in admin user registration verification is done .

Proof Of Creation of table and user registration

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| bank               |
| mydb_test          |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.00 sec)

mysql> USE BANK
Database changed
mysql> SHOW TABLES
    -> ;
+-----------------+
| Tables_in_bank  |
+-----------------+
| account_details |
| admins          |
| logs            |
| users           |
+-----------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM USERS;
+----+---------+---------------------+------------+----------+--------------+----------------+--------+-------------------------+
| ID | NAME    | EMAIL_ID            | DOB        | PASSWORD | PHONE_NUMBER | AADHAAR_NUMBER | GENDER | REGISTERED_TIMESTAMP    |
+----+---------+---------------------+------------+----------+--------------+----------------+--------+-------------------------+
|  1 | Jaydeep | [redacted]@gmail.com | 10-05-2004 |     9898 |   1234567891 | 1234-1234-1234 | male   | 12:30:07 PM  2022/03/11 |
+----+---------+---------------------+------------+----------+--------------+----------------+--------+-------------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM LOGS;
+--------+-------------+-----------------+---------+---------------------+-------------+---------+-------------------------------+-------------------------+
| LOG_ID | USER_BANK_ID | USER_ACCOUNT_ID | NAME    | EMAIL               | PREV_AMOUNT | NEW_BAL | TASK_PERFORMED                | TIMESTAMP               |
+--------+-------------+-----------------+---------+---------------------+-------------+---------+-------------------------------+-------------------------+
|      1 |           1 |               0 | Jaydeep | [redacted]@gmail.com |           0 | 0       | USER REGISTRATION SUCCESSFULLY | 12:24:32 PM  2022/03/11 |
+--------+-------------+-----------------+---------+---------------------+-------------+---------+-------------------------------+-------------------------+
1 row in set (0.00 sec)

mysql>
```

Mysql Database & Table Strucutre

1. Database Structure

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| bank               |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

Main Database

2. Tables In Database 'bank'

```
mysql> show tables;
+-----------------+
| Tables_in_bank  |
+-----------------+
| account_details |
| admins          |
| logs            |
| users           |
+-----------------+
4 rows in set (0.00 sec)
```

3. Individual Table Structure

   i. Table : **account_details**

```
mysql> desc account_details;
+------------------------------+--------------+------+-----+---------+----------------+
| Field                        | Type         | Null | Key | Default | Extra          |
+------------------------------+--------------+------+-----+---------+----------------+
| TABLE_ID                     | int(255)     | NO   | PRI | NULL    | auto_increment |
| ID                           | int(255)     | NO   |     | NULL    |                |
| EMAIL_ID                     | varchar(50)  | NO   | UNI | NULL    |                |
| PASSWORD                     | int(20)      | NO   |     | NULL    |                |
| ACCOUNT_BALANCE              | bigint(255)  | NO   |     | NULL    |                |
| ACCOUNT_TYPE                 | char(20)     | NO   |     | NULL    |                |
| ACCOUNT_REGISTERED_TIMESTAMP | varchar(100) | YES  |     | NULL    |                |
+------------------------------+--------------+------+-----+---------+----------------+
7 rows in set (0.01 sec)
```

   ii. Table : **admins**

```
mysql> desc admins;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| ID       | int(255)    | NO   | PRI | NULL    | auto_increment |
| NAME     | char(50)    | NO   |     | NULL    |                |
| EMAIL    | varchar(50) | NO   | UNI | NULL    |                |
| PASSWORD | int(30)     | NO   |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

   iii. Table : **logs**

```
mysql> desc logs;
+-----------------+-------------+------+-----+-------------+-----------------+
| Field           | Type        | Null | Key | Default     | Extra           |
+-----------------+-------------+------+-----+-------------+-----------------+
| LOG_ID          | int(255)    | NO   | PRI | NULL        | auto_increment |
| USER_BANK_ID    | int(255)    | NO   |     | NULL        |                 |
| USER_ACCOUNT_ID | int(255)    | NO   |     | NULL        |                 |
| NAME            | char(20)    | NO   |     | NULL        |                 |
| EMAIL           | varchar(50) | NO   |     | NULL        |                 |
| PREV_AMOUNT     | int(255)    | NO   |     | NULL        |                 |
| NEW_BAL         | char(255)   | YES  |     | NOT UPDATED |                 |
| TASK_PERFORMED  | tinytext    | NO   |     | NULL        |                 |
| TIMESTAMP       | tinytext    | NO   |     | NULL        |                 |
+-----------------+-------------+------+-----+-------------+-----------------+
9 rows in set (0.00 sec)
```

iv. Table : **users**

```
mysql> desc users;
+----------------------+--------------+------+-----+---------+----------------+
| Field                | Type         | Null | Key | Default | Extra          |
+----------------------+--------------+------+-----+---------+----------------+
| ID                   | int(255)     | NO   | PRI | NULL    | auto_increment |
| NAME                 | char(50)     | NO   |     | NULL    |                |
| EMAIL_ID             | varchar(50)  | NO   | UNI | NULL    |                |
| DOB                  | varchar(100) | NO   |     | NULL    |                |
| PASSWORD             | int(20)      | NO   |     | NULL    |                |
| PHONE_NUMBER         | bigint(255)  | NO   |     | NULL    |                |
| AADHAAR_NUMBER       | varchar(50)  | NO   | UNI | NULL    |                |
| GENDER               | char(20)     | NO   |     | NULL    |                |
| REGISTERED_TIMESTAMP | varchar(100) | NO   |     | NULL    |                |
+----------------------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

Query To Create Tables In Database

1. account_details :

```
CREATE TABLE IF NOT EXISTS ACCOUNT_DETAILS(TABLE_ID INT(255) NOT NULL PRIMARY KEY
AUTO_INCREMENT,ID INT(255) NOT NULL,EMAIL_ID VARCHAR(50) NOT NULL UNIQUE,PASSWORD INT(20)
NOT NULL,ACCOUNT_BALANCE BIGINT(255) NOT NULL, ACCOUNT_TYPE CHAR(20) NOT
NULL,ACCOUNT_REGISTERED_TIMESTAMP VARCHAR(100));
```

2. admins :

```
CREATE TABLE IF NOT EXISTS ADMINS(ID INT(255) NOT NULL PRIMARY KEY AUTO_INCREMENT,NAME
CHAR(50) NOT NULL,EMAIL VARCHAR(50) NOT NULL UNIQUE,PASSWORD INT(30) NOT NULL);
```

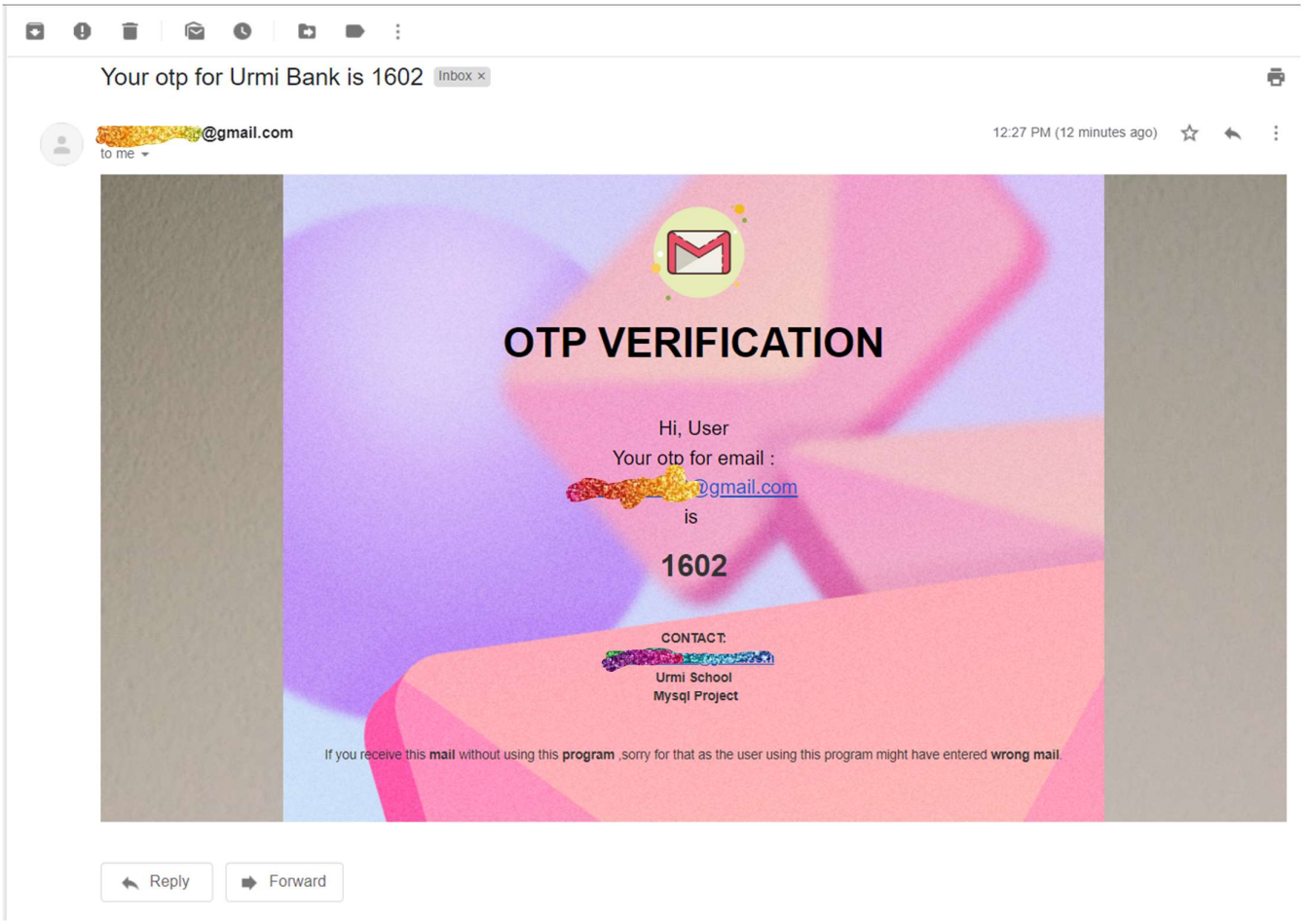
3. logs :

```
CREATE TABLE IF NOT EXISTS LOGS(LOG_ID INT(255) NOT NULL PRIMARY KEY
AUTO_INCREMENT,USER_BANK_ID INT(255) NOT NULL,USER_ACCOUNT_ID INT(255) NOT NULL,NAME
CHAR(20) NOT NULL,EMAIL VARCHAR(50) NOT NULL,PREV_AMOUNT INT(255) NOT NULL,NEW_BAL CHAR(255)
DEFAULT 'NOT UPDATED',TASK_PERFORMED TINYTEXT NOT NULL,TIMESTAMP TINYTEXT NOT NULL);
```

4. users :

```
CREATE TABLE IF NOT EXISTS USERS(ID INT(255) NOT NULL PRIMARY KEY AUTO_INCREMENT,NAME
CHAR(50) NOT NULL,EMAIL_ID VARCHAR(50) NOT NULL UNIQUE,DOB VARCHAR(100) NOT NULL,PASSWORD
INT(20) NOT NULL,PHONE_NUMBER BIGINT(255) NOT NULL,AADHAAR_NUMBER VARCHAR(50) NOT NULL
UNIQUE,GENDER CHAR(20) NOT NULL,REGISTERED_TIMESTAMP VARCHAR(100) NOT NULL );
```

How Email Looks?



Bibliography

1. https://www.freecodecamp.org/news/send-emails-using-code-4fcea9df63f/

   I used this to figure how to send email

2. https://stripo.email/

   I used this for creating email interface