Software Engineering

Assignment 4

Name – Trisanjit Das

Department - Information Technology

Roll No - 002311001089

Section - A3

The program code is as follows:

1. Inventory Management System with Git

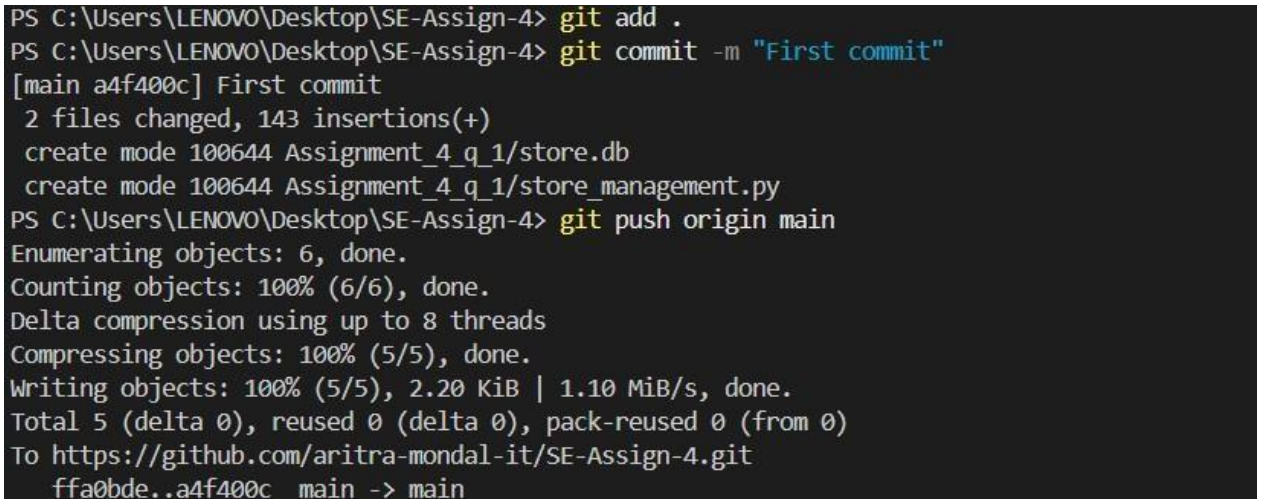
a) Design a system to manage products for a store. Customers can make purchases, and sellers can update the list of products.

b) Use Git for version control and maintain a purchase history of items.

```
import sqlite3


# Connect to SQLite database (or create it if it doesn't exist)

conn = sqlite3.connect('store.db') cursor = conn.cursor()


# Create tables if they don't exist cursor.execute('''

CREATE TABLE IF NOT EXISTS products (     id

INTEGER PRIMARY KEY AUTOINCREMENT,     name

TEXT NOT NULL,     price REAL NOT

NULL,     quantity INTEGER NOT NULL

)

''')


cursor.execute('''

CREATE TABLE IF NOT EXISTS purchases (     id

INTEGER PRIMARY KEY AUTOINCREMENT,

product_id INTEGER NOT NULL,     quantity

INTEGER NOT NULL,     total_price REAL NOT

NULL,

   FOREIGN KEY (product_id) REFERENCES products (id)

)
''')

conn.commit()

# Function to add a new product def
add_product(name, price, quantity):
    cursor.execute('''
    INSERT INTO products (name, price, quantity)
    VALUES (?, ?, ?)
    ''', (name, price, quantity))     conn.commit()
print(f"Product '{name}' added successfully!")

# Function to update a product def update_product(product_id,
name=None, price=None, quantity=None):
    if name:
        cursor.execute('''
UPDATE        products
SET name = ?
        WHERE id = ?         ''',
(name, product_id))      if
price:
        cursor.execute('''
UPDATE products         SET
price = ?
        WHERE id = ?         ''',
(price,  product_id))       if
quantity:
        cursor.execute('''         UPDATE
products         SET quantity = ?
        WHERE id = ?
        ''', (quantity, product_id))     conn.commit()
print(f"Product ID {product_id} updated successfully!")


# Function to display all products def
display_products():
cursor.execute('SELECT * FROM products')
products = cursor.fetchall()     for product in
products:

        print(f"ID: {product[0]}, Name: {product[1]}, Price: ₹{product[2]:.2f},
Quantity: {product[3]}")


# Function to make a purchase def
make_purchase(product_id, quantity):
    cursor.execute('SELECT price, quantity FROM products WHERE id = ?',
(product_id,))     product =
cursor.fetchone()     if product:

        price, available_quantity = product         if
available_quantity >= quantity:
total_price = price * quantity
cursor.execute('''
            INSERT INTO purchases (product_id, quantity, total_price)
            VALUES (?, ?, ?)
            ''', (product_id, quantity, total_price))
cursor.execute('''
            UPDATE products
SET quantity = quantity - ?
            WHERE id = ?
            ''', (quantity, product_id))             conn.commit()
print(f"Purchase successful! Total price: ${total_price:.2f}")
        else:              print("Insufficient quantity
available!")     else:          print("Product not
found!")


# Function to display all purchases def
display_purchases():
cursor.execute('''
    SELECT purchases.id, products.name,
purchases.quantity, purchases.total_price      FROM
purchases

    JOIN products ON purchases.product_id = products.id

    ''')

    purchases = cursor.fetchall()

for purchase in purchases:

        print(f"Purchase ID: {purchase[0]}, Product: {purchase[1]}, Quantity:
{purchase[2]}, Total Price: ${purchase[3]:.2f}")



# Main menu def

main():     while

True:          print("\nStore Management

System")        print("1. Add Product")

print("2.

Update Product")        print("3. Display Products")


        print("4. Make Purchase")

print("5. Display Purchases")        print("6.

Exit")        choice = input("Enter your

choice: ")



        if choice == '1':

            name = input("Enter product name: ")

price = float(input("Enter product price: "))
```

```
quantity = int(input("Enter product quantity: "))

add_product(name, price, quantity)        elif choice ==


'2':

          product_id = int(input("Enter product ID to update: "))

name = input("Enter new name (leave blank to skip): ")           price =

input("Enter new price (leave blank to skip): ")           quantity =

input("Enter new quantity (leave blank to skip): ")

          update_product(product_id, name or None, float(price) if price else
None, int(quantity) if quantity else None)

       elif choice == '3':

display_products()        elif choice

== '4':

          product_id = int(input("Enter product ID to purchase: "))

quantity = int(input("Enter quantity to purchase: "))

make_purchase(product_id, quantity)


       elif choice == '5':

          display_purchases()

elif choice == '6':

          break        else:

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



if __name__ == "__main__":

   main()
```

# Close the database connection when done conn.close()



```
PS C:\Users\LENOVO\Desktop\SE-Assign-4> git add .
PS C:\Users\LENOVO\Desktop\SE-Assign-4> git commit -m "First commit"
[main a4f400c] First commit
 2 files changed, 143 insertions(+)
 create mode 100644 Assignment_4_q_1/store.db
 create mode 100644 Assignment_4_q_1/store_management.py
PS C:\Users\LENOVO\Desktop\SE-Assign-4> git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 2.20 KiB | 1.10 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/aritra-mondal-it/SE-Assign-4.git
   ffa0bde..a4f400c  main -> main
```

2. Marks Management System with Git

a) Develop a Student Marks Management System using Git.

b) In this system, a central database stores students' marks for different subjects in a tabular format.

c) Subject teachers can update marks as needed before the final submission.

d) Teachers can view student names and roll numbers but only edit the marks for their subject.

e) When all teachers have completed their updates, the database is sorted by total marks and made available for students to view.

import sqlite3

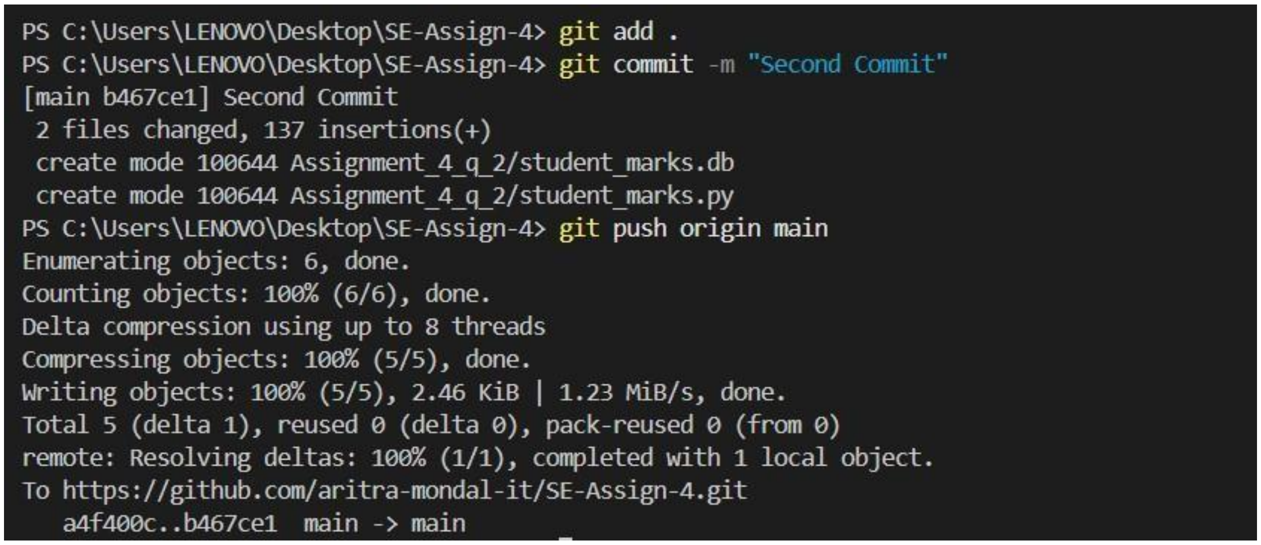
# Connect to the SQLite database (or create it if it doesn't exist)

conn = sqlite3.connect('student_marks.db') cursor = conn.cursor()

```
# Create tables if they don't exist cursor.execute('''

CREATE TABLE IF NOT EXISTS students (

roll_number INTEGER PRIMARY KEY,     name

TEXT NOT NULL

)

''')


cursor.execute('''

CREATE TABLE IF NOT EXISTS subjects (     subject_id

INTEGER PRIMARY KEY AUTOINCREMENT,

subject_name TEXT NOT NULL,     teacher_name TEXT

NOT NULL

)

''')


cursor.execute('''

CREATE TABLE IF NOT EXISTS marks (

roll_number INTEGER,     subject_id

INTEGER,     marks INTEGER,

    PRIMARY KEY (roll_number, subject_id),

    FOREIGN KEY (roll_number) REFERENCES students (roll_number),

    FOREIGN KEY (subject_id) REFERENCES subjects (subject_id)
)
''')

conn.commit()

# Function to add a new student def
add_student(roll_number, name):
    cursor.execute('''
    INSERT INTO students (roll_number, name)
VALUES (?, ?)     ''', (roll_number, name))
conn.commit()     print(f"Student '{name}' added
successfully!")

# Function to add a new subject def
add_subject(subject_name, teacher_name):
    cursor.execute('''
    INSERT INTO subjects (subject_name, teacher_name)
    VALUES (?, ?)
    ''', (subject_name, teacher_name))     conn.commit()
print(f"Subject '{subject_name}' added successfully!")

# Function to update marks for a subject def
update_marks(roll_number, subject_id, marks):
    cursor.execute('''
    INSERT OR REPLACE INTO marks (roll_number, subject_id, marks)
VALUES (?, ?, ?)
    ''', (roll_number, subject_id, marks))     conn.commit()
print(f"Marks updated for Roll Number {roll_number} in Subject ID
{subject_id}!")


# Function to view students and their roll numbers def
view_students():
    cursor.execute('SELECT * FROM students')
students = cursor.fetchall()     for student in students:
        print(f"Roll Number: {student[0]}, Name: {student[1]}")

# Function to view marks for a specific subject def
view_marks(subject_id):
    cursor.execute('''
    SELECT students.roll_number, students.name, marks.marks
    FROM marks
    JOIN students ON marks.roll_number = students.roll_number
    WHERE marks.subject_id = ?
    ''', (subject_id,))
marks = cursor.fetchall()
for mark in marks:
        print(f"Roll Number: {mark[0]}, Name: {mark[1]}, Marks: {mark[2]}")   # Function to calculate
and sort students by total marks def sort_by_total_marks():

    cursor.execute('''
    SELECT students.roll_number, students.name, SUM(marks.marks) AS
total_marks
    FROM marks
    JOIN students ON marks.roll_number = students.roll_number
    GROUP BY students.roll_number
    ORDER BY total_marks DESC
    ''')
    results = cursor.fetchall()     for
result in results:
        print(f"Roll Number: {result[0]}, Name: {result[1]}, Total Marks:
{result[2]}")


# Main menu def
main():     while
True:
        print("\nStudent Marks Management System")
print("1. Add Student")         print("2. Add Subject")         print("3.
Update Marks")         print("4. View
Students")         print("5. View Marks for a Subject")
print("6. Sort Students by Total Marks")         print("7.
Exit")
```

```
        choice = input("Enter your choice: ")


        if choice == '1':

           roll_number = int(input("Enter Roll Number: "))

name = input("Enter Student Name: ")

add_student(roll_number, name)

        elif choice == '2':

           subject_name = input("Enter Subject Name: ")

teacher_name = input("Enter Teacher Name: ")

add_subject(subject_name, teacher_name)

        elif choice == '3':

           roll_number = int(input("Enter Roll Number: "))

subject_id = int(input("Enter Subject ID: "))            marks

= int(input("Enter Marks: "))            update_marks(roll_number,

subject_id, marks)

        elif choice == '4':

view_students()        elif choice

== '5':

           subject_id = int(input("Enter Subject ID: "))

view_marks(subject_id)        elif choice == '6':

sort_by_total_marks()        elif choice == '7':


           break        else:

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

```
if __name__ == "__main__":

    main()


# Close the database connection when done conn.close()
```



```
PS C:\Users\LENOVO\Desktop\SE-Assign-4> git add .
PS C:\Users\LENOVO\Desktop\SE-Assign-4> git commit -m "Second Commit"
[main b467ce1] Second Commit
 2 files changed, 137 insertions(+)
 create mode 100644 Assignment_4_q_2/student_marks.db
 create mode 100644 Assignment_4_q_2/student_marks.py
PS C:\Users\LENOVO\Desktop\SE-Assign-4> git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 2.46 KiB | 1.23 MiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/aritra-mondal-it/SE-Assign-4.git
   a4f400c..b467ce1  main -> main
```

3. Task Management CLI Tool:

a) **Develop a command-line task management tool where users can add, edit, and complete tasks.**

b) **Implement version control to track task changes and provide a task history.**

```
import datetime
```

```
# pssd means password, ussnm is username

def user_information(ussnm, pssd):

            name = input("Enter your name please:

")          address = input("Your address: ")       age =

input("Your age please: ")    ussnm_ = ussnm+"

task.txt"           f = open(ussnm_, 'a')

            f.write(pssd)

            f.write("\nName: ")

            f.write(name)

            f.write('\n')

            f.write("Address :")


            f.write(address)

            f.write('\n')

            f.write("Age :")

            f.write(age)

            f.write('\n')

            f.close()



def signup():
```

```
            print("Please enter the username by which you wanna access your
account")         username = input("Please enter here: ")        password =
input("Enter a password: ")    user_information(username, password)    print("Sir
please proceed towards log in")
            login()



def login():
            print("Please enter your username ")
user_nm = input("Enter here: ")


            # Password as entered while logging in
pssd_wr = (input("Enter the password: "))+'\n'
            try:
                        usernm = user_nm+" task.txt"
            f_ = open(usernm, 'r')



                        # variable 'k' contains the password as saved
                        # in the file    k = f_.readlines(0)[0]
                        f_.close()
```

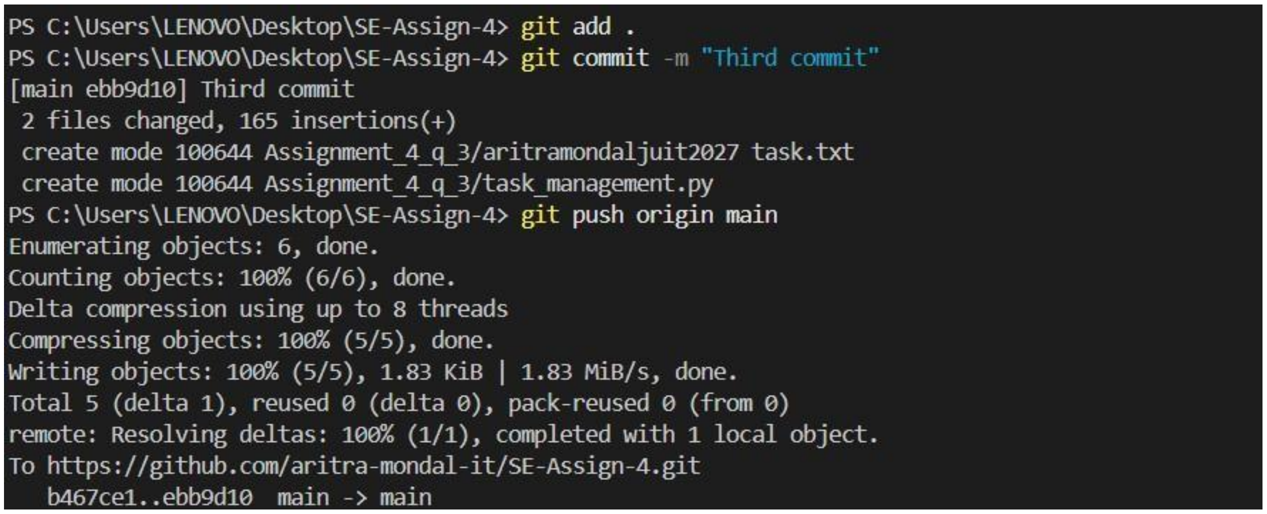
```
                    # Checking if the Password entered is same as

                    # the password saved while signing in

 if pssd_wr == k:

                              print(

                                        "1--to view your data \n2--To add task
\n3--Update task\

                                        \n4--VIEW TASK STATUS")

                    a = input()



                              if a == '1':
                                        view_data(usernm)

          elif a == '2':

                                        # add task

                                        task_information(usernm)

          elif a == '3':

                                        task_update(user_nm)
                    elif a == '4':

 task_update_viewer(user_nm)


                              else:

                                        print("Wrong input !")            else:

          print("SIR YOUR PASSWORD OR USERNAME IS WRONG")

                              login()
```

```
            except Exception as e:

                        print(e)

            login()



def view_data(username):    ff

= open(username, 'r')

print(ff.read())             ff.close()



def task_information(username):

            print("Sir enter n.o of task you want to

ADD")  j = int(input())              f1 = open(username,

'a')

            for i in range(1, j+1):

                        task = input("Enter the task : ")

target = input("Enter the target : ")    pp = "TASK

"+str(i)+' :'                      qq = "TARGET "+str(i)+" :"

                        f1.write(pp)
```

```
 f1.write(task)            f1.write('\n')     f1.write(qq)             f1.write(target)

           f1.write('\n')                          print("Do u want to stop press space

bar otherwise enter : ")

                     s = input()

           if s == ' ':

                                 break

           f1.close()




def task_update(username):

           username = username+" TASK.txt"    print("Please enter

the tasks which are completed : ")     task_completed

= input()



           print("Enter task which are still not started by you : ")

           task_not_started = input()



           print("Enter task which you are doing : ")

task_ongoing = input()


           fw  =  open(username,  'a')      DT  =

str(datetime.datetime.now())
```

```
fw.write(DT)                    fw.write("\n")

fw.write("COMPLETED        TASK        \n")

fw.write(task_completed)    fw.write("\n")

fw.write("ONGOING          TASK          \n")

fw.write(task_ongoing)       fw.write("\n")

fw.write("NOT         YET         STARTED\n")

fw.write(task_not_started)

fw.write("\n")



def task_update_viewer(username):

            ussnm = username+" TASK.txt" o

            = open(ussnm, 'r') print(o.read())

            o.close()



if __name__ == '__main__':  print("WELCOME TO ARITRA`S TASK

MANAGER")    print("sir are you new to this software")         a =

int(input("Type 1 if new otherwise press 0 ::"))

            if a == 1:

                        signup()

            elif a == 0:
```

```
            login()

        else:

            print("You have provided wrong input !")
```



```
PS C:\Users\LENOVO\Desktop\SE-Assign-4> git add .
PS C:\Users\LENOVO\Desktop\SE-Assign-4> git commit -m "Third commit"
[main ebb9d10] Third commit
 2 files changed, 165 insertions(+)
 create mode 100644 Assignment_4_q_3/aritramondaljuit2027 task.txt
 create mode 100644 Assignment_4_q_3/task_management.py
PS C:\Users\LENOVO\Desktop\SE-Assign-4> git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 1.83 KiB | 1.83 MiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/aritra-mondal-it/SE-Assign-4.git
   b467ce1..ebb9d10  main -> main
```

Please find the GitHub link attached with this to go through my GitHub account repository -

https://github.com/TrisanjitrisingSD/SE_Lab_2025_A3_4_Repo.git