SUNBEAM SCHOOL VARUNA

VARANASI

Computer Science (New) (083)

Project On

Python and MySQL

(Book Shop Management System)

Session 2020-21

under the guidance of: -

Ms. Ruhi Seth

Made by:-

Swarit Jain

(XII-A,)

Tanishq Singh

(XII-C,)

Acknowledgement

We would like to convey our thanks to ***Ms. Ruhi Seth***, our ***COMPUTER SCIENCE*** teacher, Sunbeam School, Varuna for her immense help and guidance in the completion of our project. It is only due to her effort that our project could be completed successfully. This report is submitted as a part of practical examination included in curriculum of CBSE for ***AISSE*** (All India Senior Secondary Examination) to be held in the year 2020-21.

SUNBEAM SCHOOL VARUNA

Certificate

This is to certify that **Swarit Jain (Class 12-A)** and **Tanishq Singh (Class 12-C)** of class XII of Sunbeam School Varuna, Varanasi has completed this project under my supervision. They have taken interest and shown utmost sincerity in completion of this project. They have successfully completed the project work in 'Computer Science (083)' up to my satisfaction.

(P.G.T. Computer Science)

Index

Source Code

Outputs

What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- Web development (Server-side),
- Software development,
- Mathematics,
- System scripting

I) What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

II) Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

- Python can be treated in a procedural way, an object-oriented way or a functional way.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.

III) Good to know

- The most recent major version of Python is Python 3. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- Python comes with a pre-built text editor. It is also possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, NetBeans or Eclipse which are particularly useful when managing larger collections of Python files.

IV) Python Syntax compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

What is MySQL ?

- MySQL is a database system used for developing web-based software applications.
- MySQL used for both small and large applications.
- MySQL is a relational database management system *(RDBMS).*
- MySQL is fast, reliable, and flexible and easy to use.
- MySQL supports standard SQL *(Structured Query Language).*
- MySQL is free to download and use.
- MySQL was developed by Michael Widenius and David Axmark in 1994.
- MySQL is presently developed, distributed, and supported by Oracle Corporation.
- MySQL Written in C, C++.

Main Features of MySQL

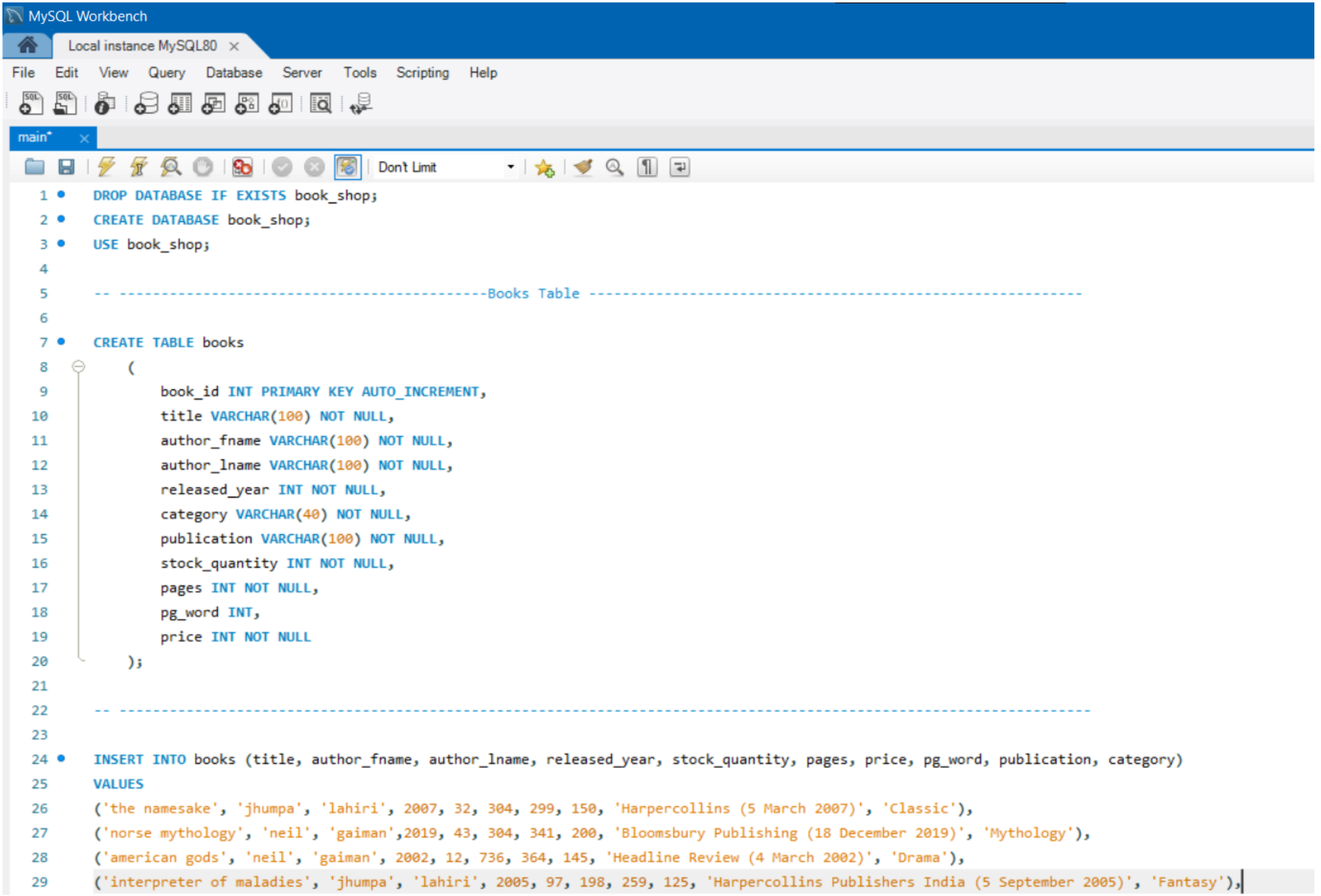
- MySQL server design is multi-layered with independent modules.
- MySQL is fully multithreaded by using kernel threads. It can handle multiple CPUs if they are available.
- MySQL provides transactional and non-transactional storage engines.
- MySQL has a high-speed thread-based memory allocation system.

- MySQL Handles large databases.
- MySQL Server works in client/server or embedded systems.
- MySQL Works on many different platforms.
- MySQL supports in memory heap table

Who Uses MySQL

- Some of the most famous websites like Facebook, Wikipedia, Google (not for search), YouTube, Flickr.
- Content Management Systems (CMS) like WordPress, Drupal, Joomla, phpBB etc.
- A large number of web developers worldwide are using MySQL to develop web applications.

Complete MySQL Data with Test Entries




```sql
DROP DATABASE IF EXISTS book_shop;
CREATE DATABASE book_shop;
USE book_shop;

-- ----------------------------------------------Books Table ----------------------------------------------------------

CREATE TABLE books
    (
        book_id INT PRIMARY KEY AUTO_INCREMENT,
        title VARCHAR(100) NOT NULL,
        author_fname VARCHAR(100) NOT NULL,
        author_lname VARCHAR(100) NOT NULL,
        released_year INT NOT NULL,
        category VARCHAR(40) NOT NULL,
        publication VARCHAR(100) NOT NULL,
        stock_quantity INT NOT NULL,
        pages INT NOT NULL,
        pg_word INT,
        price INT NOT NULL
    );

-- ------------------------------------------------------------------------------------------------------------------

INSERT INTO books (title, author_fname, author_lname, released_year, stock_quantity, pages, price, pg_word, publication, category)
VALUES
('the namesake', 'jhumpa', 'lahiri', 2007, 32, 304, 299, 150, 'Harpercollins (5 March 2007)', 'Classic'),
('norse mythology', 'neil', 'gaiman',2019, 43, 304, 341, 200, 'Bloomsbury Publishing (18 December 2019)', 'Mythology'),
('american gods', 'neil', 'gaiman', 2002, 12, 736, 364, 145, 'Headline Review (4 March 2002)', 'Drama'),
('interpreter of maladies', 'jhumpa', 'lahiri', 2005, 97, 198, 259, 125, 'Harpercollins Publishers India (5 September 2005)', 'Fantasy'),
```

main*

Don't Limit

```
('a hologram for the king: a novel', 'dave', 'eggers', 2013, 25, 154, 315, 129, 'Penguin UK (1 August 2013)', 'Travel'),
('the circle', 'dave', 'eggers', 2014, 26, 504, 395, 140, 'Penguin (24 April 2014)', 'Horror'),
('the amazing adventures of kavalier & clay', 'michael', 'chabon', 2012, 68, 701, 499, 165, 'Random House Trade Paperbacks (12 June 2012)', 'Adventure'),
('just kids', 'patti', 'smith', 2010, 55, 304, 259, 250, 'Ecco (2 November 2010)', 'Fairy Tale'),
('a heartbreaking work of staggering genius', 'dave', 'eggers', 2007, 104, 437, 431, 160, 'Picador (21 September 2007)', 'Humour'),
('coraline', 'neil', 'gaiman', 2016, 100, 208, 216, 175, "Bloomsbury Children's Books (30 August 2016)'", 'Mystery'),
('what we talk about when we talk about love: stories', 'raymond', 'carver', 2003, 23, 176, 301, 190, 'Vintage Classics (5 November 2009)', 'Love Story'),
("where i'm calling from: selected stories", 'raymond', 'carver', 1989, 12, 526, 487, 200, 'Vintage (18 June 1989)', 'Thriller'),
('white noise', 'don', 'delillo', 2017, 49, 320, 699, 180, 'Penguin Classics (18 January 2017)', 'Thriller'),
('cannery row', 'john', 'steinbeck', 2011, 95, 181, 434, 300, 'Penguin UK (16 May 2011)', 'Short Story'),
('oblivion: stories', 'david', 'foster wallace', 2005, 172, 329, 299, 200, 'Abacus (28 April 2005)', 'Suspense'),
('consider the lobster', 'david', 'foster wallace', 2007, 92, 343, 582, 230, 'Abacus (21 June 2007)', 'Collection of Essays'),
('10% happier', 'dan', 'harris', 2017, 29, 256, 485, 140, 'Yellow Kite (26 January 2017)', 'Romance'),
('lincoln in the bardo', 'george', 'saunders', 2017, 1000, 367, 100, 250, 'Bloomsbury Publishing (1 January 2017)', 'Drama');

ALTER TABLE books engine = InnoDB;

-- ----------------------------------------------Orders Table ----------------------------------------------------------
CREATE TABLE orders
    (ord_no INT PRIMARY KEY auto_increment,    -- 0
     customer_name VARCHAR(40) NOT NULL,       -- 1
     address VARCHAR(100) NOT NULL,            -- 2
     district VARCHAR(30) NOT NULL,            -- 3
     city VARCHAR(30) NOT NULL,                -- 4
     state_ut VARCHAR(20) NOT NULL,            -- 5
     pincode INT NOT NULL,                     -- 6
     country VARCHAR(25) NOT NULL DEFAULT 'India', -- 7
     e_mail VARCHAR(80) NOT NULL CHECK(e_mail LIKE '%@%'),  -- 8
     phone_no BIGINT(10),    -- 9
```

main*

Don't Limit

```
59      pieces INT NOT NULL,    -- 10
60      bookname VARCHAR(100) NOT NULL,   -- 11
61      order_dt TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- 12
62      user_name VARCHAR(30) NOT NULL, -- 13
63      order_type CHAR(7) NOT NULL CHECK(order_type in ('replace', 'new', 'return')), -- 14
64      FOREIGN KEY (ord_no) REFERENCES books(book_id)
65     );
66
67  ALTER TABLE orders engine = InnoDB;
68
69  INSERT INTO orders (ord_no, customer_name, address, district, city, state_ut, pincode, country, e_mail, phone_no, pieces, bookname, order_dt, user_name, order_type)
70  VALUES
71  ('Swarit', 'C-121, SBI Colony, Main Branch Campus, Opposite Kachahari', 'Varanasi', 'Varanasi', 'Uttar Pradesh', 221002, 'India',
72  'swaritjain123@gmail.com', 8004124465, 5, 'american gods', '2020-10-26 15:30:11', 'admin', 'new'),
73  ('Swarit', 'C-121, SBI Colony, Main Branch Campus, Opposite Kachahari', 'Varanasi', 'Varanasi', 'Uttar Pradesh', 221002, 'India',
74  'swaritjain123@gmail.com', 8004124465, 1, 'coraline', '2020-11-01 15:30:11', 'admin', 'new'),
75  (5, 'Swarit', 'C-121, SBI Colony, Main Branch Campus, Opposite Kachahari', 'Varanasi', 'Varanasi', 'Uttar Pradesh', 221002, 'India',
76   'swaritjain123@gmail.com', 8004124465, 2, 'the namesake', '2020-11-02 15:30:11', 'admin', 'return');
77
78   -- Orders inserted above are being used in the program to show outputs of replace and return functions with the help of date difference b/w 7 to 14 days
79
80  -- ------------------------------------------ Accounts Table ------------------------------------------------------------
81  CREATE TABLE accounts
82      (ID BIGINT PRIMARY KEY AUTO_INCREMENT,
83       username VARCHAR(30) UNIQUE NOT NULL,
84       passwd VARCHAR(40) NOT NULL,
85       name_u VARCHAR(50) NOT NULL,
86       email VARCHAR(50) NOT NULL CHECK(email LIKE '%@%') UNIQUE
87       );
```

```sql
-- ---------------------------------------------- Test Accounts ------------------------------------------------------

INSERT INTO accounts (username, passwd, name_u, email)
VALUES
('admin', 'admin', 'admin', 'swaritjain123@gmail.com'),
('testuser', 'testuser', 'test', 'tanishq.rc@gmail.com');

ALTER TABLE accounts engine = InnoDB;
-- ---------------------------------------------------------------------------------------------------------------------
```

books.py

File Edit View Navigate Code Refactor Run Tools VCS Window Help main_menu.py - books.py

Class 12 Computer Project > books.py >

books.py

```python
import mysql.connector as conn1
import string
#
''' This function will help the administrator to add as many books they want in the 'books' table '''

def Add_book():
    mycon1 = conn1.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
    cursor = mycon1.cursor(buffered=True)
    no_books = int(input("Enter number of books you want to add: "))
    for i in range(no_books):
        try:
            book_name = str(input("     Enter the book name: ")).lower()
            writer_fname = str(input("     Enter the first name of author: ")).lower()
            writer_lname = str(input("     Enter the last name of author: ")).lower()
            year = int(input("     Enter the year of Release: "))
            cat = str(input("     Enter the category of book: ")).lower()
            pub = str(input("     Enter the Publisher's Name: ")).lower()
            stock = int(input("     Enter the number of books in stock: "))
            page = int(input("     Enter number of pages of book: "))
            words = int(input("     Enter the number of words per page (average) approximate: "))
            price = int(input("     Enter the price of the book: "))
            cursor.execute(
                f"INSERT INTO books (title, author_fname, author_lname, released_year, stock_quantity, category, "
                f"publication, pages, pg_word, price) VALUES('{book_name}', '{writer_fname}', '{writer_lname}', {year}, "
                f"{stock}, '{cat}', '{pub}', {page}, {words}, {price})")
            mycon1.commit()
            print("Successfully Added the book.")
        except:
            print("Wrong Data Given.")
            Add_book()
    mycon1.close()
    return
```

1: Project 7: Structure 2: Favorites AWS Explorer

books.py

```
#
''' This function will help the administrator to update books' stock quantity,
    price, number of pages, release year '''

def Update_book():
    mycon1 = conn1.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
    cursor = mycon1.cursor(buffered=True)
    no_books = int(input("Enter number of books you want to update: "))
    count = 0
    while count < no_books:
        book_name = str(input("Enter the name of the book you want to update: ")).lower()
        cursor.execute("SELECT * FROM books WHERE title = '{}'".format(book_name))
        data = cursor.fetchone()
        if data == None:
            print("No book like '{}' exists in database.".format(book_name))  # Checking if the book exists in database
            Update_book()                                                     # If not found, function runs again.
        else:
            print("     1. Update the book's year: \n",
                  "     2. Update the book's stock: \n",
                  "     3. Update the book's number of pages: \n"
                  "     4. Update the book's price: ")
            choice = int(input("Enter your choice: "))
            if choice == 1:
                year = int(input("Enter the new year of the book: "))
                cursor.execute("UPDATE books SET released_year = {} where title = '{}'".format(year, book_name))
                mycon1.commit()  # Updating the book's release year
                print("Successfully Updated")
            elif choice == 2:
                stock = int(input("Enter the stock of the book: "))
                cursor.execute("UPDATE books SET stock_quantity = {} where title = '{}'".format(stock, book_name))
                mycon1.commit()  # Updating the book's stock quantity
                print("Successfully Updated")
```

books.py

```
            elif choice == 3:
                page = int(input("Enter the number of pages of the book: "))
                cursor.execute("UPDATE books SET pages = {} where title = '{}'".format(page, book_name))
                mycon1.commit()  # Updating the book's number of pages as per new release
                print("Successfully Updated")
            elif choice == 4:
                price_n = int(input("Enter the new price: "))
                cursor.execute("UPDATE books SET price = {} where title = '{}'".format(price_n, book_name))
                mycon1.commit()  # Updating the book's price
                print("Successfully Updated")
            else:
                print("Wrong Input")
                Update_book()
            count += 1
    mycon1.close()
    return

#
''' This function will help the administrator to delete books from the books table'''

def Delete_book():
    mycon1 = conn1.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
    cursor = mycon1.cursor(buffered=True)
    no_books = int(input("Enter number of books you want to delete: "))
    count = 0
    while count < no_books:
        book_name = str(input("Enter the name of the book you want to delete: ")).lower()
        cursor.execute("SELECT * FROM books WHERE title = '{}'".format(book_name))
        data = cursor.fetchone()
        if data == None:
            print("No book like '{}' exists in database.".format(book_name))  # Checking if the book exists in database
```

```
97          else:
98              try:
99                  cursor.execute("DELETE FROM books where title = '{}'".format(book_name))
100                 mycon1.commit()  # Deleting the book
101                 print("Successfully Deleted.")
102             except:
103                 print("This book can't be deleted because Orders for this book exists.")
104         count += 1
105     mycon1.close()
106     return
107
108 #
109 ''' This function will help administrator to view every detail of as many books admin wants.'''
110
111 def View_details():
112     mycon1 = conn1.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
113     cursor = mycon1.cursor(buffered=True)
114     no_books = int(input("Enter number of books of which you want details of?: "))
115     count = 0
116     while count < no_books:
117         book_name = str(input("Enter the name of the book " + str(count + 1) + ': ')).lower()
118         cursor.execute("SELECT * FROM books WHERE title = '{}'".format(book_name))
119         data = cursor.fetchone()
120         if data == None:
121             choice = str(input(
122                 f"No book like '{book_name}' exists in database. Renter Again? (Y or N): "))  # Checking the Wrong Input
123             if choice == 'Y' or 'y':
124                 View_details()
125             else:
126                 exit(0)
127         else:
128             print("    Book ID:                ", data[0])
```

books.py

```
127         else:
128             print("    Book ID:                  ", data[0])
129             print("    Book Name:                ", string.capwords(data[1]))
130             print("    Author's Name:            ", string.capwords(data[2]) + ' ' + string.capwords(data[3]))
131             print("    Release Year:             ", data[4])
132             print("    Category:                 ", string.capwords(data[5]))
133             print("    Publication:              ", string.capwords(data[6]))
134             print("    Pages:                    ", data[8])
135             print("    Words Per Page (approx): ", data[9])
136             print("    Stock Quantity:           ", data[7])
137             print("    Price:                    ", data[10])
138         count += 1
139     mycon1.close()
140     return
141
142 #
143 '''This function will help administrator to view details of every book in the shop'''
144
145 def View_all():
146     mycon1 = conn1.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
147     cursor = mycon1.cursor(buffered=True)
148     cursor.execute("SELECT * FROM books")
149     data = cursor.fetchall()
150     for row in data:
151         tp = row   # fetchall return tuple of tuples. Therefore, taking one tuple at a time in 'tp' through for loop.
152         print("    Book ID:                  ", tp[0])
153         print("    Book Name:                ", string.capwords(tp[1]))
154         print("    Author's Name:            ", string.capwords(tp[2]) + ' ' + string.capwords(tp[3]))
155         print("    Release Year:             ", tp[4])
156         print("    Category:                 ", string.capwords(tp[5]))
157         print("    Publication:              ", string.capwords(tp[6]))
158         print("    Pages:                    ", tp[8])
```

```
159         print("    Words Per Page (approx): ", tp[9])
160         print("    Stock Quantity:          ", tp[7])
161         print("    Price:                   ", tp[10])
162         print()
163     mycon1.close()
164     return
165
166 # ------------------------------------------------------------------------------------------------
167  # |                                  End of Module                                              |
168 # ------------------------------------------------------------------------------------------------
```

orders.py

orders.py

```python
# Inbuilt Modules
import smtplib
from string import capwords
from time import sleep
import datetime

# Project Modules
import mysql.connector as mycam
import config


# ------------------------------------------------------------------------------------------------ #
# |                            Below Functions are Common                                        | #
# ------------------------------------------------------------------------------------------------ #

# ---------------------------------------------------------------------
''' This function will allow every user in database to place order for any book if it exists in database.
    User is prompted every time to place another order if he/she wants.
    When user says 'n' or 'no' function are called from other modules to generate invoice and invoice number.'''

def Place_ord(username):
    ord_id_ls = []
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    cursor.execute("SET SQL_SAFE_UPDATES = 0")

    def work():
        print()
        book_name = str(input('Enter the name of the book ' + ': ')).lower()
        cursor.execute("SELECT * FROM books WHERE title='{}'".format(book_name))
        data = cursor.fetchone()
        if data == None:
```

```
data = cursor.fetchone()
if data == None:
    print('Sorry, No book named', book_name, 'found.')
else:
    print(f"   Book named , {capwords(book_name)},  found details are as follows:-")
    print("    Author's name:           ", capwords(data[2]) + ' ' + capwords(data[3]))
    print("    Released Year:           ", data[4])
    print("    Number of Pages:         ", data[8])
    print("    Words Per Page (Approx): ", data[9])
    print("    Publication:             ", capwords(data[6]))
    print("    Category of Book:        ", capwords(data[5]))
    print("    Available quantity:      ", data[7])
    print("    Price of the book:       ", data[10])
    choice = str(input("Do you want to order this Book ? "))

    if choice in ['yes', 'Yes', 'YES', 'y', 'Y']:
        quantity = int(input('Enter the QUANTITY you want to order: '))
        print('              Checking Stock, please wait...')
        sleep(2)

        if quantity <= data[7]:    # Checking if pieces specified to be bought are present or not for the specified book.
            print('              Enough quantity found')
            print('Please fill up your details below: ')

            name = str(input('     Enter Name: '))
            address = str(input('     Enter Address: '))
            dist = str(input('     Enter District: '))
            city = str(input('     Enter City: '))
            stu = str(input('     Enter State/UT: '))
            pin = int(input("     Enter Pincode: "))
            country = str(input('     Enter Country: '))
            phone = int(input('     Enter Phone number: '))
```

```
63
64          # Email Address is fetched from the accounts table. This email address will be used to send email
65          cursor.execute(f"SELECT email FROM accounts where username = '{username}'")
66          email = cursor.fetchone()
67          amount = data[10] * quantity
68          cursor.execute(f"SELECT ID FROM accounts WHERE username = '{username}'")
69          custom_id = cursor.fetchone()
70          print('  Now we came at the final step of the process...')
71          print("             Total Amount to be paid is: ", amount)
72          print("             No Delivery Charges Applicable, i.e. Delivery Free")
73          confirm = str(input('  Are you sure you want to confirm this order? '))
74
75          if confirm in ['yes', 'Yes', 'YES', 'y', 'Y']:
76              details = [name, address, dist, city, stu, pin, country, phone, custom_id]
77              cursor.execute(
78                  f"UPDATE books SET stock_quantity = stock_quantity - {quantity} WHERE title = '{book_name}'")
79              cam.commit()
80              cursor.execute(
81                  f"INSERT INTO orders "
82                  f"(customer_name, address, district, city, state_ut, pincode, country, e_mail, phone_no, pieces, bookname, "
83                  f"user_name, order_type) "
84                  f"VALUES ('{name}', '{address}', '{dist}', '{city}', '{stu}', {pin}, '{country}', '{email[0]}', {phone}, {quantity}, "
85                  f"'{book_name}', '{username}', 'new')")
86              cam.commit()
87              cursor.execute(
88                  f"SELECT ord_no, order_dt FROM orders WHERE customer_name = '{name}' and phone_no = {phone} ORDER BY ord_no DESC")
89              data2 = cursor.fetchone()
90              ord_id_ls.append(data2[0])
91              print("Processing your order....")
92              sleep(2)
93              print()
94              print("               Order Placed Successfully.")
```

```
                        print()
                        print("              Order Placed Successfully.")
                    else:
                        print('Your order has been discarded')

                else:
                    print('Sorry', data[7], 'pieces are left')
                    print('Thanks for visiting, keep shopping :)')
            elif choice in ['no', 'No', 'NO', 'n', 'N']:
                print('Okay.')
            else:
                print('Invalid Input')

        print()
        ch = str(input("Want to place more orders? (y/n): ")).lower()
        if ch in ['y', 'yes']:
            work()
        else:
            details.append(email[0])
            details.append(username)
            inv = config.Inv_no_gen(username)
            config.Pdf_generate(details, ord_id_ls, inv[0], inv[1])
            print("Thanks for visiting us :)")
            print()
    work()
    return

# ---------------------------------------------------------------------
'''This function will be used only if the order is specified for return/replace or order is cancelled
   For that, message and subject or email is defined in their respective functions.
   Sender's Email Address and Password is given in config module of the project
   An email will be sent to the email address of the user with the details.'''
```

```
125
126 def Send_email(email, subject, msg):
127     try:
128         server = smtplib.SMTP('smtp.gmail.com:587')
129         server.ehlo()
130         server.starttls()
131         server.login(config.EMAIL_ADDRESS, config.PASSWORD)
132         final_message = 'Subject: {}\n\n{}'.format(subject, msg)
133         server.sendmail(config.EMAIL_ADDRESS, email, final_message)
134         server.quit()
135         print(f"An email with all details is sent to you on your E-mail Address {email}")
136     except:
137         print("Email failed to send.")
138     return
139
140 # -----------------------------------------------------------------
141 '''This function will allow the user to return or replace the order within 7 days after the delivery date.
142    Else a message will be displayed that the order is not eligible for passing the Return/Replace Policy.
143    By default, the delivery date is the date 7 days after the date of placing the order'''
144
145 def Return_replace(username):
146     cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
147     cursor = cam.cursor(buffered=True)
148     cursor.execute(f"SELECT ord_no FROM orders WHERE user_name = '{username}'")
149     record = cursor.fetchall()
150     L = []
151     if not record:
152         print('No order to return or replace')
153     else:
154         cursor.execute(
155             f"SELECT * from orders where user_name = '{username}' and DATEDIFF(NOW(), order_dt) >= 7 and DATEDIFF(NOW(), order_dt) <= 14 "
156             f"ORDER by ord_no DESC")
```

```
data = cursor.fetchall()
if not data:
    print('None of your order is eligible to return or replace.')
else:
    print('We have found', len(data), 'orders have been delivered to you')
    print('List of the orders that have been delivered to you are as follows:-')
    for i in data:
        L.append(i[0])
        print('   Book Name      : ', i[11])
        print('   Order Date     : ', i[12])
        print('   Order ID       : ', i[0])
        print('   Customer Name  : ', i[1])
        print(f"   Location       :  {i[2]}, {i[3]}, {i[4]}, {i[5]}, {i[6]}, {i[7]}")
        print('   Email          : ', i[8])
        print('   Phone no       : ', i[9])
        print('   No of Books    : ', i[10])
        print()

    choice = str(input('Do you want to return or replace any of the above orders (y/n) ')).lower()

    def work(c):
        if c == 'y':
            ID = int(input('Enter the order ID for which you want to return or replace: '))
            if ID not in L:
                print('No order ID ', ID, ' matched from the orders that have been delivered to you')
            else:
                print('You have chosen order ID: ', ID)
                print('Few details of your order are as follows: ')
                print()
                cursor.execute(f"SELECT * FROM orders WHERE ord_no = {ID}")
                data2 = cursor.fetchone()
                print('   Book name: ', data2[11])
```

```
        print('   Book Name      : ', i[11])
        print('   Order Date     : ', i[12])
        print('   Order ID       : ', i[0])
        print('   Customer Name  : ', i[1])
        print(f"   Location       :  {i[2]}, {i[3]}, {i[4]}, {i[5]}, {i[6]}, {i[7]}")
        print('   Email          : ', i[8])
        print('   Phone no       : ', i[9])
        print('   No of Books    : ', i[10])
        print()

choice = str(input('Do you want to return or replace any of the above orders (y/n) ')).lower()

def work(c):
    if c == 'y':
        ID = int(input('Enter the order ID for which you want to return or replace: '))
        if ID not in L:
            print('No order ID ', ID, ' matched from the orders that have been delivered to you')
        else:
            print('You have chosen order ID: ', ID)
            print('Few details of your order are as follows: ')
            print()
            cursor.execute(f"SELECT * FROM orders WHERE ord_no = {ID}")
            data2 = cursor.fetchone()
            print('   Book name: ', data2[11])
            print('   Quantity: ', data2[10])
            print('   Order date: ', data2[12])
            print()
            print('By SNT.bookshop Return and Replace policies, you have below two options')
            print('1. Return')
            print('2. Replace')

            opt = int(input('Enter the option number that you want to select: '))
```

```
197
198                 if opt == 1:
199                     print('You have entered under our Return policy')
200                     confirm = str(input('Are you sure you want to return this order? (y/n) ')).lower()
201                     if confirm in ['y', 'yes']:
202                         print('   Processing the Return')
203                         msg = f"Your request for returning your order has been accepted. Some details for it are as follows: \n" \
204                               f"Book Name: {capwords(data2[11])} \n" \
205                               f"Order ID: {data2[0]} \n" \
206                               f"Name: {data2[1]} \n" \
207                               f"Address: {data2[2]} \n" \
208                               f"District: {data2[3]} \n" \
209                               f"City: {data2[4]} \n" \
210                               f"State: {data2[5]} \n" \
211                               f"Pincode: {data2[6]} \n" \
212                               f"Country: {data2[7]} \n" \
213                               f"Phone: {data2[9]} \n" \
214                               f"Pieces Ordered: {data2[10]} \n" \
215                               f"Order Date & Time: {data2[12]} \n"
216                         sub = f"Information of Returning Order on S & T Bookshop"
217                         Send_email(data2[8], sub, msg)  # Calling Function to send email
218                         # Real Email is sent here.
219                         # Email is sent actually to the mailbox if the given email really exists
220                         cursor.execute(
221                             f"INSERT INTO orders "
222                             f"(customer_name, address, district, city, state_ut, pincode, country, e_mail, phone_no, pieces, bookname, "
223                             f"user_name, order_type) VALUES ('{data2[1]}', '{data2[2]}', '{data2[3]}', '{data2[4]}', '{data2[5]}', "
224                             f"{data2[6]}, '{data2[7]}', '{data2[8]}', {data2[9]}, {data2[10]}, "
225                             f"'{data2[11]}', '{username}', 'return')")
226                         cam.commit()
227                         cursor.execute(f"DELETE FROM orders WHERE ord_no = {ID}")
228                         cam.commit()
```

```
                    print('   Successfully requested for return to your order.')
                    print(
                        '   Our courier boy will come within the next day to collect the package after verifying its condition,
                        '   be ready with the packed book')
                    print('   Money will be refund to your account within 2-3 days')
                    return
                else:
                    print('   Return process cancelled')

            elif opt == 2:
                print('You have entered under our Replace policy.')
                confirm = str(input('Are you sure you want to replace for this order? (y/n) ')).lower()

                if confirm in ['y', 'yes']:
                    msg = f"Your request for replacing your order has been accepted. Some details for it are as follows: \n" \
                          f"Book Name: {capwords(data2[11])} \n" \
                          f"Order ID: {data2[0]} \n" \
                          f"Name: {data2[1]} \n" \
                          f"Address: {data2[2]} \n" \
                          f"District: {data2[3]} \n" \
                          f"City: {data2[4]} \n" \
                          f"State: {data2[5]} \n" \
                          f"Pincode: {data2[6]} \n" \
                          f"Country: {data2[7]} \n" \
                          f"Phone: {data2[9]} \n" \
                          f"Pieces Ordered: {data2[10]} \n" \
                          f"Order Date & Time: {data2[12]} \n" \
                          f"Delivery will be provided within 7 working days. \n" \
                          f"Sorry for the inconvenience :)"
                    sub = f"Information of Replacing Order on S & T Bookshop"
                    print("   Successfully requested for replacement of your order.")
                    Send_email(data2[8], sub, msg)  # Calling Function to send email
```

```
                    send_email(data2[0], sub, msg)  # calling function to send email
                    # Real Email is sent here.
                    # Email is sent actually to the mailbox if the given email really exists
                    cursor.execute(
                        f"INSERT INTO orders "
                        f"(customer_name, address, district, city, state_ut, pincode, country, e_mail, phone_no, pieces, bookname, "
                        f"user_name, order_type) VALUES ('{data2[1]}', '{data2[2]}', '{data2[3]}', '{data2[4]}', '{data2[5]}', {data2[6]}, "
                        f"'{data2[7]}', '{data2[8]}', {data2[9]}, {data2[10]}, "
                        f"'{data2[11]}', '{username}', 'replace')")
                    cam.commit()
                    cursor.execute(f"DELETE FROM orders WHERE ord_no = {ID}")
                    cam.commit()
                    print(
                        '   Our courier boy will come within the next day to collect the package after verifying its condition, '
                        '   be ready with the packed book')
                    print(
                        "   Replacement will be completed within 7 working days until your order has been delivered. ")
                elif confirm in ['n', 'no']:
                    print("Replacement Cancelled")
                    print("Exiting")
                    return
                else:
                    print('Invalid option choosen.')
            elif c in ['n', 'no']:
                print("Exiting")
                return
            else:
                print("Wrong Input")
                again_t = str(input("Do you want to return or replace any of the above orders? (y/n) ")).lower()
                if again_t in ['yes', 'y']:
                    work(again_t)
```

```
                    elif again_t in ['n', 'no']:
                        print("Exiting")
                        return
                    else:
                        print("Invalid Input.")

            work(choice)
        cam.close()
        return


# ------------------------------------------------------------------------------------------------ #
# |                            Below Functions are for Administrator                               |
# ------------------------------------------------------------------------------------------------ #

# ---------------------------------------------------------------------
'''This function will allow the administrator to cancel as many orders he wants by specifying the Order ID
   An email will be sent to the user with details. Order can be cancelled only within 7 days after placing order'''

def Cancel_ord():  # Admin can cancel any user's order
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    cursor.execute("SELECT ord_no FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7")
    rec = cursor.fetchall()
    if len(rec) == 0:
        print("No orders to cancel.")
        print("Sorry")
    else:
        print(f"Only {len(rec)} order(s) is/are eligible to cancel.")
        num = int(input(f"Enter number of orders you want to cancel (not more than {len(rec)}): "))
        if num <= len(rec):
```

```
321         if num <= len(rec):
322             for i in range(num):
323                 order_id = int(input('  Enter the Order ID: '))
324                 cursor.execute(f"SELECT ord_no FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {order_id}")
325                 data3 = cursor.fetchone()
326                 if len(data3) == 1:
327                     cursor.execute(f"SELECT * FROM orders WHERE ord_no = {order_id}")
328                     data = cursor.fetchone()
329                     if data == None:
330                         print(f"Order ID {order_id} does not exist.")
331                     else:
332                         cursor.execute("SET SQL_SAFE_UPDATES = 0")
333                         cursor.execute(
334                             f"UPDATE books SET stock_quantity = stock_quantity + {data[10]} WHERE title = '{data[11]}'")
335                         cam.commit()
336                         cursor.execute(f"SELECT publication, author_fname, author_lname FROM books WHERE title = '{data[11]}'")
337                         data2 = cursor.fetchone()
338                         msg = f"Your Order has been cancelled as per your request. Please review the product on our website. \n" \
339                               f"Book Name: {capwords(data[11])} \n" \
340                               f"Publication: {capwords(data2[0])} \n" \
341                               f"Author's Name: {capwords(data2[1])} {capwords(data2[2])} \n" \
342                               f"Order ID: {data[0]} \n" \
343                               f"Name: {capwords(data[1])} \n" \
344                               f"Address: {data[2]} \n" \
345                               f"District: {data[3]} \n" \
346                               f"City: {data[4]} \n" \
347                               f"State: {data[5]} \n" \
348                               f"Pincode: {data[6]} \n" \
349                               f"Country: {data[7]} \n" \
350                               f"Phone: {data[9]} \n" \
351                               f"Pieces Ordered: {data[10]} \n" \
352                               f"Order Date & Time: {data[12]}"
```

```
                                f"Pieces Ordered: {data[10]} \n" \
                                f"Order Date & Time: {data[12]}"
                            email = data[8]
                            sub = 'Regarding Cancellation of Order on S & T Bookshop'
                            Send_email(email, sub, msg)
                            cursor.execute(f"DELETE FROM orders WHERE ord_no = {order_id}")
                            cam.commit()
                            print("Successfully cancelled your order.")
                        else:
                            print("Order not eligible to cancel.")
                    else:
                        print(f"Not possible to cancel more than {len(rec)} orders.")
                        Cancel_ord()
            cam.close()
            return

# ---------------------------------------------------------------------
'''This function will allow administrator to update the order address of multiple orders'''

def Update_ord_addr():  # Admin can update any user's order
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    cursor.execute("SELECT * FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7")
    lim = cursor.fetchall()
    if not lim:
        print("No orders are eligible for update of address.")
    else:
        print(f"Only {len(lim)} order(s) is/are eligible for update of address.")

        def work(data):
            num = int(input('Enter the number of orders\'s addresses you want to update: '))
            lst = [(data[i][0]) for i in range(len(data))]
```

```
383         if num <= len(data):
384             for i in range(num):
385                 order_id = int(input('Enter the Order ID to update the order: '))
386                 if order_id in lst:
387                     new_addr = str(input('     Enter the new address for order ID ' + str(order_id) + ': '))
388                     new_city = str(input('     Enter the new city for order ID    ' + str(order_id) + ': '))
389                     new_pinc = int(input('     Enter the new pincode for order ID ' + str(order_id) + ': '))
390                     new_dist = str(input('     Enter the new district for Order ID' + str(order_id) + ': '))
391                     new_stat = str(input('     Enter the new State/UT for Order ID' + str(order_id) + ': '))
392                     cursor.execute("SET SQL_SAFE_UPDATES = 0")
393                     cursor.execute("UPDATE orders SET address = '{}' WHERE ord_no = {}".format(new_addr, order_id))
394                     cam.commit()
395                     cursor.execute("UPDATE orders SET city = '{}'  WHERE ord_no = {}".format(new_city, order_id))
396                     cam.commit()
397                     cursor.execute("UPDATE orders SET pincode = {}  WHERE ord_no = {}".format(new_pinc, order_id))
398                     cam.commit()
399                     cursor.execute("UPDATE orders SET district = '{}'  WHERE ord_no = {}".format(new_dist, order_id))
400                     cam.commit()
401                     cursor.execute("UPDATE orders SET state_ut = '{}'  WHERE ord_no = {}".format(new_stat, order_id))
402                     cam.commit()
403                     cursor.execute(f"SELECT e_mail FROM orders WHERE ord_no = {order_id}")
404                     email = cursor.fetchone()
405                     msg = f"Your Order has been updated successfully \n. " \
406                           f"New Location of the order is {new_addr}, {new_city}, {new_pinc}, {new_city}, {new_stat} :)"
407                     sub = "Information of Order Update on S & T Book Shop Management System"
408                     Send_email(email[0], sub, msg)
409                     print('Successfully Updated')
410                     print(f"regarding order update for new location on email address {email[0]}")
411                 else:
412                     print("This order ID is not eligible for update of order address.")
413                     continue
414         else:
```

```
                continue
            else:
                print("Invalid Input. Try Again")
                work(data)
        work(lim)
    cam.close()
    return

# ---------------------------------------------------------------------
'''This function will allow administrator to view all orders in database of every user.'''

def View_all_ords():  # Admin can view all user's order
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    print("  1 -> View all New Orders Placed \n"
          "  2 -> View all Orders for Return \n"
          "  3 -> View all Orders for Replacement \n"
          "  4 -> View all Orders including all categories")
    ch = int(input("  Enter your choice (1 to 4): "))

    def start(choice):
        if choice == 1:
            cursor.execute("SELECT * FROM orders WHERE order_type = 'new' ORDER BY ord_no DESC")
            data1 = cursor.fetchall()
            if not data1:
                print("No orders.")
            else:
                for row in data1:
                    tp = row
                    print("    Order ID       : ", tp[0])
                    print("    Book Name      : ", capwords(tp[11]))
                    print("    Pieces         : ", tp[10])
```

```
445                     print("    Customer Name  : ", capwords(tp[1]))
446                     print("    Address        : ", tp[2])
447                     print("    District       : ", tp[3])
448                     print("    City           : ", tp[4])
449                     print("    State          : ", tp[5])
450                     print("    Pincode        : ", tp[6])
451                     print("    Country        : ", tp[7])
452                     print("    E-mail Address : ", tp[8])
453                     print("    Phone          : ", tp[9])
454                     print("    Order Date     : ", tp[12])
455                     print("    Username       : ", tp[13])
456                     print()
457         elif choice == 2:
458             cursor.execute("SELECT * FROM orders WHERE order_type = 'return' ORDER BY ord_no DESC")
459             data2 = cursor.fetchall()
460             if not data2:
461                 print("No orders.")
462             else:
463                 for row in data2:
464                     tp = row
465                     print("    Order ID       : ", tp[0])
466                     print("    Book Name      : ", capwords(tp[11]))
467                     print("    Pieces         : ", tp[10])
468                     print("    Customer Name  : ", capwords(tp[1]))
469                     print("    Address        : ", tp[2])
470                     print("    District       : ", tp[3])
471                     print("    City           : ", tp[4])
472                     print("    State          : ", tp[5])
473                     print("    Pincode        : ", tp[6])
474                     print("    Country        : ", tp[7])
475                     print("    E-mail Address : ", tp[8])
476                     print("    Phone          : ", tp[9])
```

```
                print("    Order Date     : ", tp[12])
                print("    Username       : ", tp[13])
                print()
        elif choice == 3:
            cursor.execute("SELECT * FROM orders WHERE order_type = 'replace' ORDER BY ord_no DESC")
            data1 = cursor.fetchall()
            if not data1:
                print("No orders.")
            else:
                for row in data1:
                    tp = row
                    print("    Order ID       : ", tp[0])
                    print("    Book Name      : ", capwords(tp[11]))
                    print("    Pieces         : ", tp[10])
                    print("    Customer Name  : ", capwords(tp[1]))
                    print("    Address        : ", tp[2])
                    print("    District       : ", tp[3])
                    print("    City           : ", tp[4])
                    print("    State          : ", tp[5])
                    print("    Pincode        : ", tp[6])
                    print("    Country        : ", tp[7])
                    print("    E-mail Address : ", tp[8])
                    print("    Phone          : ", tp[9])
                    print("    Order Date     : ", tp[12])
                    print("    Username       : ", tp[13])
                    print()
        elif choice == 4:
            cursor.execute("SELECT * FROM orders ORDER BY ord_no DESC")
            data = cursor.fetchall()
            if not data:
                print("No orders.")
            else:
```

```
            for row in data:
                tp = row
                print("    Order ID       : ", tp[0])
                print("    Book Name      : ", capwords(tp[11]))
                print("    Pieces         : ", tp[10])
                print("    Customer Name  : ", capwords(tp[1]))
                print("    Address        : ", tp[2])
                print("    District       : ", tp[3])
                print("    City           : ", tp[4])
                print("    State          : ", tp[5])
                print("    Pincode        : ", tp[6])
                print("    Country        : ", tp[7])
                print("    E-mail Address : ", tp[8])
                print("    Phone          : ", tp[9])
                print("    Order Date     : ", tp[12])
                print("    Username       : ", tp[13])
                print("    Order Type     : ", tp[14])
                if tp[14] not in ['return', 'replace']:
                    cursor.execute(
                        f"SELECT DATE(order_dt) FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {tp[0]}")
                    data2 = cursor.fetchone()
                    if not data2:
                        cursor.execute(
                            f"SELECT DATE(DATE_ADD(order_dt, INTERVAL 7 DAY)) FROM orders WHERE ord_no = {tp[0]}")
                        data3 = cursor.fetchone()
                        print(f"    Order Status   :  Delivered on {data3[0]}")
                    else:
                        print("    Order Status   :  In transit")
                print()
        else:
            ch2 = int(input("Wrong Input. Enter Again: "))
```

```
            ch2 = int(input("Wrong Input. Enter Again: "))
            start(ch2)

    start(ch)
    cam.close()
    return

# ---------------------------------------------------------------------
'''This function will allow administrator to view all orders of a user'''

def View_all_u():  # Admin can view any user's order particularly
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    cursor.execute("SELECT ord_no FROM orders")
    data = cursor.fetchall()
    if not data:
        print("No orders in database.")
    else:
        user = str(input("Enter the customer's username: "))
        cursor.execute(f"SELECT * FROM orders WHERE user_name = '{user}' ORDER BY order_dt DESC")
        data2 = cursor.fetchall()
        if data2 == None:
            print(f"No orders of username '{user}'.")
        else:
            for row in data2:
                tp = row
                print("    Order ID       : ", tp[0])
                print("    Book Name      : ", capwords(tp[11]))
                print("    Pieces         : ", tp[10])
                print("    Customer Name  : ", capwords(tp[1]))
                print("    Address        : ", tp[2])
```

```
                print("    Address        : ", tp[2])
                print("    District       : ", tp[3])
                print("    City           : ", tp[4])
                print("    State          : ", tp[5])
                print("    Pincode        : ", tp[6])
                print("    Country        : ", tp[7])
                print("    E-mail Address : ", tp[8])
                print("    Phone          : ", tp[9])
                print("    Order Date     : ", tp[12])
                print("    Order Type     : ", capwords(tp[14]))
                if tp[14] not in ['return', 'replace']:
                    cursor.execute(
                        f"SELECT DATE(order_dt) FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {tp[0]}")
                    data2 = cursor.fetchone()
                    if not data2:
                        cursor.execute(
                            f"SELECT DATE(DATE_ADD(order_dt, INTERVAL 7 DAY)) FROM orders WHERE ord_no = {tp[0]}")
                        data3 = cursor.fetchone()
                        print(f"    Order Status   :  Delivered on {data3[0]}")
                    else:
                        print("    Order Status   :  In transit")
                print()
    cam.close()
    return

# ---------------------------------------------------------------------
'''This function will allow administrator to view all orders of a particular date'''

def View_ord_dt():  # Admin can view any user's order by date
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    cursor.execute("SELECT ord_no FROM orders")
```

```
601     data = cursor.fetchall()
602     if not data:
603         print("No orders in database.")
604     else:
605         dt = str(input("Enter date (YYYY-MM-DD): "))
606         cursor.execute(f"SELECT * FROM orders WHERE DATE(order_dt) = '{dt}'")
607         data = cursor.fetchall()
608         for row in data:
609             tp = row
610             print("    Order ID       : ", tp[0])
611             print("    Book Name      : ", capwords(tp[11]))
612             print("    Pieces         : ", tp[10])
613             print("    Customer Name  : ", capwords(tp[1]))
614             print("    Address        : ", tp[2])
615             print("    District       : ", tp[3])
616             print("    City           : ", tp[4])
617             print("    State          : ", tp[5])
618             print("    Pincode        : ", tp[6])
619             print("    Country        : ", tp[7])
620             print("    E-mail Address : ", tp[8])
621             print("    Phone          : ", tp[9])
622             print("    Order Date     : ", tp[12])
623             print("    Username       : ", tp[13])
624             print("    Order Type     : ", capwords(tp[14]))
625             if tp[14] not in ['return', 'replace']:
626                 cursor.execute(f"SELECT DATE(order_dt) FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {tp[0]}")
627                 data2 = cursor.fetchone()
628                 if not data2:
629                     cursor.execute(f"SELECT DATE(DATE_ADD(order_dt, INTERVAL 7 DAY)) FROM orders WHERE ord_no = {tp[0]}")
630                     data3 = cursor.fetchone()
631                     print(f"    Order Status   :  Delivered on {data3[0]}")
632                 else:
```

```
                print("    Order Status    :  In transit")
            print()
        cam.close()
        return

# ------------------------------------------------------------------------------------------------------------
# |                                  Below Functions are only for non-admin users                             |
# ------------------------------------------------------------------------------------------------------------

# ---------------------------------------------------------------------
'''This function will allow users to cancel their order only if they cancel it within 7 days of the date since
   order is placed. An email will be sent to the user with the details.'''

def Cancel_ord_u(username):
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    cursor.execute(
        f"SELECT * from orders where user_name = '{username}' and DATEDIFF(NOW(), order_dt) <= 7 ORDER by ord_no DESC")
    data = cursor.fetchall()
    list1 = []
    if not data:
        print("None of your orders are eligible to cancel. Sorry :)")
    else:
        print("This is the list of your orders which are eligible to cancel: ")
        for row in data:
            tp = row
            list1.append(tp[0])
            print("    Order ID       : ", tp[0])
            print("    Book Name      : ", capwords(tp[11]))
            print("    Pieces         : ", tp[10])
            print("    Customer Name  : ", capwords(tp[1]))
```

```
663             print("    Customer Name  : ", capwords(tp[1]))
664             print("    Address        : ", tp[2])
665             print("    District       : ", tp[3])
666             print("    City           : ", tp[4])
667             print("    State          : ", tp[5])
668             print("    Pincode        : ", tp[6])
669             print("    Country        : ", tp[7])
670             print("    E-mail Address : ", tp[8])
671             print("    Phone          : ", tp[9])
672             print("    Order Date     : ", tp[12])
673             print("    Order Type     : ", tp[14])
674
675             if tp[14] not in ['return', 'replace']:
676                 cursor.execute(f"SELECT DATE(order_dt) FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {tp[0]}")
677                 data2 = cursor.fetchone()
678                 if not data2:
679                     cursor.execute(f"SELECT DATE(DATE_ADD(order_dt, INTERVAL 7 DAY)) FROM orders WHERE ord_no = {tp[0]}")
680                     data3 = cursor.fetchone()
681                     print(f"    Order Status   :  Delivered on {data3[0]}")
682                 else:
683                     print("    Order Status   :  In transit")
684             print()
685
686         def choose():
687             choice = int(input("Enter the Order ID to cancel: "))
688             return choice
689
690         ch = choose()
691         if ch in list1:
692             cursor.execute(f"SELECT * FROM orders WHERE ord_no = {ch}")
693             data2 = cursor.fetchone()
694             cursor.execute(f"SELECT publication, author_fname, author_lname FROM books WHERE title = '{data2[11]}'")
```

```
        data3 = cursor.fetchone()
        msg = f"Your Order has been cancelled as per your request. Please review the product on our website. \n\n" \
              f"Book Name: {capwords(data2[11])} \n" \
              f"Publication: {capwords(data3[0])} \n" \
              f"Author's Name: {capwords(data3[1])} {capwords(data3[2])} \n" \
              f"Order ID: {data2[0]} \n" \
              f"Name: {capwords(data2[1])} \n" \
              f"Address: {data2[2]} \n" \
              f"District: {data2[3]} \n" \
              f"City: {data2[4]} \n" \
              f"State: {data2[5]} \n" \
              f"Pincode: {data2[6]} \n" \
              f"Country: {data2[7]} \n" \
              f"Phone: {data2[9]} \n" \
              f"Pieces Ordered: {data2[10]} \n" \
              f"Order Date & Time: {data2[12]} \n" \
              f"Order Status: Cancelled on {datetime.datetime.now()}"
        email = data2[8]
        sub = 'Regarding Cancellation of Order on S & T Bookshop'
        Send_email(email, sub, msg)
        cursor.execute("SET SQL_SAFE_UPDATES = 0")
        cursor.execute(
            f"UPDATE books SET stock_quantity = stock_quantity + {data2[10]} WHERE title = '{data2[11]}'")
        cam.commit()
        cursor.execute(f"DELETE FROM orders where ord_no = {ch}")
        cam.commit()
        print("Successfully Cancelled the Order.")
    if len(list1) > 1:
        ch2 = str(input("   Want to cancel more Orders? (y OR n) "))
        if ch2 == 'y' or ch2 == 'Y':
            Cancel_ord_u(username)
```

```
                Cancel_ord_u(username)
            elif ch2 == 'n' or ch2 == 'N':
                cam.close()
                return
            else:
                print("Wrong Input")
                return
        else:
            pass

# ---------------------------------------------------------------------
'''This function will allow administrator to update his/her order address if they do it within 7 days after
   order is being placed. An email will be sent with few order details and new location of delivery.'''

def Up_ord_au(username):
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    cursor.execute(f"SELECT * FROM orders WHERE user_name = '{username}' and DATEDIFF(NOW(), order_dt) <= 7 ORDER BY ord_no DESC")
    data = cursor.fetchall()
    list1 = []
    if data == None:
        print("No 'IN TRANSIT' orders in database for your username.")
    else:
        print("This is the list of your orders eligible for updating the address: ")
        for row in data:
            tp = row
            list1.append(tp[0])
            print("    Order ID:       ", tp[0])
            print("    Book Name:      ", capwords(tp[11]))
            print("    Pieces:         ", tp[10])
            print("    Customer Name:  ", capwords(tp[1]))
            print("    Address:        ", tp[2])
```

```
757         print("    District:       ", tp[3])
758         print("    City:           ", tp[4])
759         print("    State/UT:       ", tp[5])
760         print("    Pincode:        ", tp[6])
761         print("    Country:        ", tp[7])
762         print("    E-mail Address: ", tp[8])
763         print("    Phone:          ", tp[9])
764         print("    Order Date:     ", tp[12])
765         print()
766
767     def choose():
768         choice = int(input("Enter the Order ID to update address: "))
769         return choice
770     ch = choose()
771     if ch in list1:
772         new_addr = str(input("Enter the new address: "))
773         new_city = str(input("Enter City: "))
774         new_pinc = int(input('Enter pincode: '))
775         new_dist = str(input("Enter District: "))
776         new_stat = str(input("Enter State/UT" + ': '))
777         cursor.execute("SET SQL_SAFE_UPDATES = 0")
778         cursor.execute(f"UPDATE orders SET address = '{new_addr}' WHERE ord_no = {ch}")
779         cam.commit()
780         cursor.execute(f"UPDATE orders SET city = '{new_city}' WHERE ord_no = '{ch}'")
781         cam.commit()
782         cursor.execute(f"UPDATE orders SET pincode = {new_pinc}  WHERE ord_no = {ch}")
783         cam.commit()
784         cursor.execute(f"UPDATE orders SET district = '{new_dist}' WHERE ord_no = '{ch}'")
785         cam.commit()
786         cursor.execute(f"UPDATE orders SET state_ut = '{new_stat}' WHERE ord_no = '{ch}'")
787         cam.commit()
788         print('Successfully Updated')
```

```
            cursor.execute(f"SELECT email FROM accounts WHERE username = '{username}'")
            email = cursor.fetchone()
            msg = f"Your Order has been updated successfully. \n" \
                  f"New Location of the order is {new_addr}, {new_city}, {new_pinc}, {new_city}, {new_stat} :)"
            sub = "Information of Order Update on S & T Book Shop Management System"
            Send_email(email[0], sub, msg)
            print()

        if len(list1) > 1:
            ch2 = str(input("   Want to update more Orders? (y OR n): "))
            if ch2 == 'y' or ch2 == 'Y':
                Up_ord_au(username)
            elif ch2 == 'n' or ch2 == 'N':
                return
            else:
                print("Wrong Input")
                return
        else:
            pass
    cam.close()
    return

#  ---------------------------------------------------------------------
'''This function will allow the signed in user to view all their details of every order placed since
   registered with detail of order type (i.e, new or for replace or for return)'''

def View_au(username):
    cam = mycam.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor(buffered=True)
    print("This is the list of your orders.: ")
    cursor.execute(f"SELECT * FROM orders WHERE user_name = '{username}' ORDER BY ord_no DESC")
    data = cursor.fetchall()
```

orders.py

```
821     if data == None:
822         print("No orders in database.")
823     else:
824         print("                  Latest Orders Sorted by Date:. ")
825         for row in data:
826             tp = row
827             print("    Order ID       : ", tp[0])
828             print("    Book Name      : ", capwords(tp[11]))
829             print("    Pieces         : ", tp[10])
830             print("    Customer Name  : ", capwords(tp[1]))
831             print("    Address        : ", tp[2])
832             print("    District       : ", tp[3])
833             print("    City           : ", tp[4])
834             print("    State/UT       : ", tp[5])
835             print("    Pincode        : ", tp[6])
836             print("    Country        : ", tp[7])
837             print("    E-mail Address : ", tp[8])
838             print("    Phone          : ", tp[9])
839             print("    Order Date     : ", tp[12])
840             print("    Order Type     : ", capwords(tp[14]))
841
842             if tp[14] not in ['return', 'replace']:
843                 cursor.execute(f"SELECT DATE(order_dt) FROM orders WHERE DATEDIFF(NOW(), order_dt) <= 7 AND ord_no = {tp[0]}")
844                 data2 = cursor.fetchone()
845                 if not data2:
846                     cursor.execute(f"SELECT DATE(DATE_ADD(order_dt, INTERVAL 7 DAY)) FROM orders WHERE ord_no = {tp[0]}")
847                     data3 = cursor.fetchone()
848                     print(f"    Order Status   :  Delivered on {data3[0]}")
849                 else:
850                     print("    Order Status   :  In transit")
851             print()
```

```
852         cam.close()
853         return
854
855     # ------------------------------------------------------------------------------------------
856     # |                                    End of Module                                       |
857     # # ------------------------------------------------------------------------------------------
858
```

acc_ctrl.py

File Edit View Navigate Code Refactor Run Tools VCS Window Help main_menu.py - acc_ctrl.py

Class 12 Computer Project > acc_ctrl.py >

acc_ctrl.py

```python
import mysql.connector as conn2
from time import sleep
from validate_email import validate_email

# -----------------------------------------------------------------------
'''This function will help the administrator to add new users.
   Here, Username entered will be accepted only if same username is already not present in database.
   Secondly, here email entered will be accepted only if it really exists
   (i.e) the given email-address has an MX record and SMTP server port. Disposable emails will
   not be accepted here. Module used here is 'validate_email' '''

def Add_new():
    mycon2 = conn2.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
    cursor = mycon2.cursor(buffered=True)
    no_user = int(input('Enter number of user you want to add: '))
    for i in range(no_user):
        input_username = input("    Enter Login ID for the user: ")
        cursor.execute(f"SELECT username FROM accounts WHERE username = '{input_username}'")
        data = cursor.fetchone()
        if not data:
            input_password = input("    Enter Password of the user: ")
            input_name = input("    Enter name of the user: ")

            def ent():
                input_email = input("    Enter email of the user: ")
                cursor.execute(f"SELECT * FROM accounts WHERE email = '{input_email}'")
                data2 = cursor.fetchone()
                if data2 != None:
                    print("E-mail Address Already Registered.")
                    ent()
```

```
30                  ent()
31              else:
32                  print("Validating Details, please wait.....")
33                  try:
34                      is_valid = validate_email(input_email)
35                      if is_valid:
36                          cursor.execute(
37                              f"INSERT INTO accounts (username, passwd, name_u, email) VALUES('{input_username}', '{input_password}', "
38                              f"'{input_name}', '{input_email}')")
39                          mycon2.commit()
40                          print("Registering")
41                          sleep(2)
42                          print(f"Successfully Added User '{input_username}'")
43                      except:
44                          print("Some Error Occurred.")
45
46          ent()
47      else:
48          print("User already in database.")
49          continue
50      mycon2.close()
51      return
52
53  # ---------------------------------------------------------------------
54  '''This function will allow the administrator to update the every detail of the user like name, password
55     username and email-address. Here email entered will be accepted only if it really exists (i.e) the given
56     email-address has an MX record and SMTP server port. Disposable emails will not be accepted here.
57     Modules used here are 'validate_email' and 'mysql.connector' '''
58
59  def Update_user():
```

```python
def Update_user():
    mycon2 = conn2.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
    cursor = mycon2.cursor(buffered=True)
    no_user = int(input('Enter number of user you want to update: '))
    count = 0
    while count < no_user:
        user_name = str(input("    Enter the username of the user you want to update: "))
        cursor.execute("SELECT * FROM accounts WHERE username = '{}'".format(user_name))
        data = cursor.fetchone()
        if data == None:
            print(f"No user like '{user_name}' exists in database.")  # Checking the wrong Input
            Update_user()
        elif data[0] == 1:
            print("Can't Update user. The specified username is Administrator.")
        else:
            print("     1 -> Update the user's username \n",
                  "    2 -> Update the user's password \n"
                  "     3 -> Update the user's name \n"
                  "     4 -> Update the user's email address")
            choice = int(input("Enter your choice (1 to 4): "))
            if choice == 1:
                new_username = str(input(f"    Enter the new username of {user_name}: "))
                cursor.execute("SELECT * FROM accounts WHERE username = '{}'".format(new_username))
                data2 = cursor.fetchone()
                if data2 == None:
                    cursor.execute(f"UPDATE accounts SET username = '{new_username}' WHERE username = '{user_name}'")
                    mycon2.commit()  # Updating the username
                    print("Successfully Updated")
                else:
                    print("Username Already Exists. \n User Not Updated :(")
```

```
89          elif choice == 2:
90              new_password = str(input(f"    Enter the new password of the {user_name}: "))
91              cursor.execute(f"UPDATE accounts SET passwd = '{new_password}' WHERE username = '{user_name}'")
92              mycon2.commit()  # Updating the user
93              print("Successfully Updated")
94          elif choice == 3:
95              new_name = str(input(f"    Enter the new name of the {user_name}: "))
96              cursor.execute(f"UPDATE accounts SET name_u = '{new_name}' WHERE username = '{user_name}'")
97              mycon2.commit()  # Updating the user
98              print("Successfully Updated")
99          elif choice == 4:
100             new_mail = str(input(f"    Enter the new email address of the {user_name}: "))
101             cursor.execute(f"SELECT email FROM accounts WHERE email = '{new_mail}'")
102             data3 = cursor.fetchone()
103             if data3 == None:
104                 is_valid = validate_email(new_mail)
105                 if is_valid:
106                     cursor.execute(f"UPDATE accounts SET email = '{new_mail}' WHERE username = '{user_name}'")
107                     mycon2.commit()  # Updating the user
108                     print("Successfully Updated")
109                 else:
110                     print("Invalid Email Address. User not updated :(")
111             else:
112                 print("Email Address Already registered. \n User Not updated :(")
113         else:
114             print("Wrong Input")
115             Update_user()
116     count += 1
117 mycon2.close()
118 return
```

acc_ctrl.py

```
120     # ------------------------------------------------------------------------
121     ''' This function will help administrator to delete any user he/she wants from database
122         by entering his/her username if it exists. But admin can't delete the administrator account
123         Module used here is 'mysql.connector' '''
124
125     def Del_user():
126         mycon2 = conn2.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
127         cursor = mycon2.cursor(buffered=True)
128         no_user = int(input("Enter number of accounts you want to delete: "))
129         count = 0
130         while count < no_user:
131             user_name = str(input("    Enter the username of the user you want to delete: "))
132             cursor.execute(f"SELECT * FROM accounts WHERE username = '{user_name}'")
133             data = cursor.fetchone()
134             if data == None:
135                 print(f"No user like '{user_name}' exists in database.")  # Checking the Wrong Input
136                 Del_user()
137             elif data[0] == 1:
138                 print("Can't Delete user. The specified username is Administrator.")
139             else:
140                 cursor.execute(f"DELETE FROM accounts where username = '{user_name}'")
141                 mycon2.commit()  # Deleting the user
142                 sleep(2)
143                 print("Successfully Deleted.")
144             count += 1
145         mycon2.close()
146         return
147
```

```
148  # ---------------------------------------------------------------------
149  ''' This function will allow administrator to view every detail of every user in database.
150      Module used here is mysql.connector '''
151
152  def All_u():
153      mycon2 = conn2.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
154      cursor = mycon2.cursor(buffered=True)
155      cursor.execute("SELECT * FROM accounts")
156      data = cursor.fetchall()
157      count = 0
158      for row in data:
159          tp = row
160          count += 1
161          print()
162          print(f"          User {count}")
163          print("      ID             : ", tp[0])
164          print("      Username       : ", tp[1])
165          print("      Password       : ", tp[2])
166          print("      Name           : ", tp[3])
167          print("      E-mail Address : ", tp[4])
168      mycon2.close()
169      return
170
171  # ---------------------------------------------------------------------
172  '''This function will help administrator to view details of the user by entering
173     his/her username if it exists in database.'''
174
175  def View_user():
176      mycon2 = conn2.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
177      cursor = mycon2.cursor(buffered=True)
```

```
177         cursor = mycon2.cursor(buffered=True)
178         num = int(input("Enter the number of users of which you want details of: "))
179         count = num
180         while count != 0:
181             user_name = str(input("    Enter the username of the user: "))
182             cursor.execute(f"SELECT * FROM accounts WHERE username = '{user_name}'")
183             data = cursor.fetchone()
184             if not data:
185                 print("Username specified is invalid. Enter Again.")
186                 print()
187             else:
188                 print(f"   ID:            {data[0]}")
189                 print(f"   Username:      {data[1]}")
190                 print(f"   Name:          {data[3]}")
191                 print(f"   Email-Address: {data[4]}")
192                 print()
193                 count -= 1
194         return
195
196     # ---------------------------------------------------------------------
197     '''Sign_Up function will allow any new user running the software to register himself in database.
198        For this, they have to create a username, password, give their name and enter their email address which
199        really exists as it will be checked then only the user will be registered.
200        Module used here is 'validate_email' and 'mysql.connector' '''
201
202     def Sign_Up():
203         mycon2 = conn2.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
204         cursor = mycon2.cursor(buffered=True)
205         input_username = input("            Create Username (Login ID) : ")
206         cursor.execute(f"SELECT * FROM accounts WHERE username = '{input_username}'")
```

```
206     cursor.execute(f"SELECT * FROM accounts WHERE username = '{input_username}'")
207     data = cursor.fetchone()
208     if data == None:
209         input_password = input("            Create Password: ")
210         input_name = input("           Enter your name: ")
211 
212         def ent():
213             input_email = input("            Enter your email address: ")
214             cursor.execute(f"SELECT * FROM accounts WHERE email = '{input_email}'")
215             data2 = cursor.fetchone()
216             if data2 != None:
217                 print("E-mail Address Already Registered.")
218                 ent()
219             else:
220                 print("Validating Details, please wait.....")
221                 is_valid = validate_email(input_email)
222                 if is_valid:
223                     cursor.execute(
224                         f"INSERT INTO accounts (username, passwd, name_u, email) VALUES('{input_username}', "
225                         f"'{input_password}', '{input_name}', '{input_email}')")
226                     mycon2.commit()
227                     print("Registering")
228                     sleep(2)
229                     print(f"Successfully Added User '{input_username}'")
230         ent()
231     else:
232         print("User Already In Database.")
233         print("Try Again")
234         Sign_Up()
235     mycon2.close()
236     return
237 
```

config.py

config.py

```python
import mysql.connector as mycam2
from fpdf import FPDF
from pdf_mail import sendpdf
from datetime import date
from string import capwords
# ---------------------------------------------------------------------
'''Below EMAIL_ADDRESS and PASSWORD is for sending email(s) to users (constant values)'''

EMAIL_ADDRESS = 'snt.bookshop@gmail.com'
PASSWORD = 'S&T@bookshp@341'

# ---------------------------------------------------------------------
'''This function is defined to send invoice of the order placed by any user
   which will fetch the given PDF INVOICE from the specific path where all invoices generated are saved.
   Module used here is 'pdf_mail' '''

def Pdf_mailing(file_name, address):
    sender_email_address = EMAIL_ADDRESS
    receiver_email_address = f"{address}"
    sender_email_password = PASSWORD
    subject_of_email = "Information of Order Placed on S & T Book Shop"
    body_of_email = "This is an auto generated bill of supply for your ordered book(s). \nIf any error is there, please reply to us " \
                    "on our E-mail Address snt.bookshop@gmail.com." \
                    "Thanks for shopping. :)"
    filename = file_name
    location_of_file = "D:/Swarit/Class 12/Class 12 Computer Project/invoices"
    k = sendpdf(sender_email_address, receiver_email_address,
                sender_email_password,
                subject_of_email,
                body_of_email,
                filename, location_of_file)
    k.email_send()
```

```
'''This function do two works. First, it makes a string sum of ASCII values of the username of the user
   Second it trims the string generated to length of 10 characters, query of the numbers of orders placed
   by the user since registered and store it in another string which is displayed in invoice number field
   of invoice generated on placing order.
   Module used here is 'mysql.connector'. '''

def Inv_no_gen(a):
    cam = mycam2.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
    cursor = cam.cursor()
    st = ''
    st2 = ''
    for char in a:
        x = ord(char)
        st += str(x)  # String sum of ASCII characters of username passed into function generated in st

    # Underneath 'for' loop is defined to limit the number of characters of the username to 10 so that
    # when invoice number will be displayed in the invoice it doesn't go out of the cell defined for it.
    # This is stored in 'st2'
    if len(st) <= 10:
        st2 += st
    else:
        for i in range(0, 10):
            st2 += st[i]

            # This query is done so as to get the total number of orders placed since the user has registered
    cursor.execute(f"SELECT count(ord_no) FROM orders WHERE user_name = '{a}'")
    ord_count = cursor.fetchone()
    # We are adding the result of the above query to above generated 'st2' and adding 1 to it
    # so that the invoice number generated is unique every time and no invoice gets replaced
    # due to same name
    st2 += f'{ord_count[0] + 1}'
    return st, st2
```

config.py

```
69
70  # ---------------------------------------------------------------------
71  '''This function generates PDF invoice with all the details of the user and the
72     book shop and save it on local desktop. User is asked whether he/she wants to get invoice on his/her
73     email-address or not. Invoice will be sent to the user's email address only if the user says 'y'.
74     Invoice name is string sum of all the ASCII values of the characters of username of the user and the last digit
75     will be one more than the number of all orders placed since the user has registered himself/herself.
76     This is done so that the invoice generated last time do not get replaced upon new order placed on local desktop.
77     Modules used here are 'fpdf' and 'mysql.connector'.'''
78
79  def Pdf_generate(ls1, ls2, inv, inv_no):
80      lst_sum = []
81      cam = mycam2.connect(host='localhost', user='root', passwd='Rinshu@03', database='book_shop')
82      cursor = cam.cursor()
83      cursor.execute(f"SELECT count(ord_no) FROM orders WHERE user_name = '{ls1[10]}'")
84      ord_count = cursor.fetchone()
85      pdf = FPDF()
86      pdf.add_page()
87
88      pdf.set_font("Arial", 'B', size=15)
89
90      pdf.cell(130, 5, 'S & T BOOKSHOP', 0, 0, 'L')
91      pdf.cell(60, 5, 'Bill of Supply/ Invoice', 0, 1, 'C')
92
93      pdf.set_font("Arial", '', size=12)
94
95      pdf.cell(130, 5, 'Street Address: K-61/74, Sector 8, Pandeypur', 0, 0, 'L')
96      pdf.cell(60, 5, '', 0, 1, 'L')
97
98      pdf.cell(130, 5, 'Varanasi, India, ZIP-221002', 0, 0, 'L')
99      pdf.cell(25, 5, 'Date: ', 0, 0, 'L')
100     pdf.cell(34, 5, f'{date.today()}', 0, 1, 'L')
```

```
102     pdf.cell(130, 5, 'Phone: (+91)8004125336', 0, 0, 'L')
103     pdf.cell(25, 5, 'Invoice #', 0, 0, 'L')
104     pdf.cell(34, 5, f"{inv_no}", 0, 1, 'L')
105
106     pdf.cell(130, 5, 'Fax: [+ 91 542 14512]', 0, 0, 'L')
107     pdf.cell(25, 5, 'Customer ID', 0, 0, 'L')
108     pdf.cell(34, 5, f'{ls1[8][0]}', 0, 1, 'L')
109
110     pdf.cell(130, 5, 'GSTIN: 22AABCU9603R1ZX', 0, 1, 'L')
111
112     pdf.cell(189, 10, '', 0, 1, 'L')
113
114     pdf.set_font("Arial", 'B', size=15)
115
116     pdf.cell(100, 5, 'Billing Address / Shipping Address: ', 0, 1)
117
118     pdf.set_font('Arial', '', size=12)
119
120     pdf.cell(10, 5, '', 0, 0)
121     pdf.cell(90, 5, f'{ls1[0]}', 0, 1)  # Name
122
123     pdf.cell(10, 5, '', 0, 0)
124     pdf.cell(90, 5, f'{ls1[1]}, {ls1[3]}', 0, 1)  # Address 1
125
126     pdf.cell(10, 5, '', 0, 0)
127     pdf.cell(90, 5, f'{ls1[2]}, {ls1[4]}', 0, 1)  # Address 2
128
129     pdf.cell(10, 5, '', 0, 0)
130     pdf.cell(90, 5, f'{ls1[5]}, {ls1[6]}', 0, 1)  # Pincode
131
132     pdf.cell(10, 5, '', 0, 0)
133     pdf.cell(90, 5, f'Phone: {ls1[7]}', 0, 1)  # Phone
```

```
135     pdf.cell(189, 10, '', 0, 1, 'L')
136
137     pdf.set_font('Arial', 'B', size=12)
138
139     pdf.cell(10, 5, 'S.no', 1, 0, 'C')
140     pdf.cell(100, 5, 'Description', 1, 0, 'C')
141     pdf.cell(25, 5, 'Quantity', 1, 0, 'C')
142     pdf.cell(25, 5, 'Rate', 1, 0, 'C')
143     pdf.cell(30, 5, 'Amount(Rs.)', 1, 1, 'C')
144
145     pdf.set_font('Arial', '', 12)
146
147     # For Loop defined below is to enter the details(quantity, rate, bookname, author's name and amount)
148     # into the invoice (i.e.) one invoice will be generated until an unless user exits the function.
149     # The last provided address and phone will be written in the billing address of thr invoice
150
151     for i in range(len(ls2)):
152         cursor.execute(f"SELECT bookname, pieces FROM orders WHERE ord_no = {ls2[i]}")
153         data = cursor.fetchone()
154         cursor.execute(f"SELECT author_fname, author_lname, price FROM books WHERE title = '{data[0]}'")
155         data2 = cursor.fetchone()
156         pdf.cell(10, 5, f"{i + 1}", 1, 0, 'R')
157         pdf.cell(100, 5, f'{capwords(data[0])} - {capwords(data2[0])} {capwords(data2[1])}', 1, 0, align='L')
158         pdf.cell(25, 5, f'{data[1]}', 1, 0, align='R')
159         pdf.cell(25, 5, f'{data2[2]}', 1, 0, align='R')
160         pdf.cell(30, 5, f'{data[1] * data2[2]}', 1, 1, align='R')
161         lst_sum.append(data[1] * data2[2])
162     # Here total of all the books ordered will be calculated
163     total = 0
164     for num in lst_sum:
165         total += num
166
```

```
166
167         pdf.cell(10, 5, '', 0, 0)
168         pdf.cell(100, 5, '', 0, 0)
169         pdf.cell(25, 5, '', 0, 0)
170
171         pdf.set_font('Arial', 'B', size=12)
172
173         pdf.cell(25, 5, 'Total', 1, 0, 'C')
174
175         pdf.set_font('Arial', 'B', size=12)
176         # Total calculated above will be displayed here
177         pdf.cell(30, 5, f'{total}', 1, 1, 'R')
178
179         file_n = f"invoice{inv}{ord_count[0] + 1} "
180         path = f"D:/Swarit/Class 12/Class 12 Computer Project/invoices/{file_n}.pdf"
181         pdf.output(f"{path}")  # Invoice generated and saved to local desktop
182
183         choice = str(input("    Do you want the invoice to be sent to your email address? (y/n): ")).lower()
184         if choice in ['y', 'yes']:
185             Pdf_mailing(file_n, address=ls1[9])
186             print(f"An email has been sent to you with an invoice to your email address {ls1[9]}")
187         else:
188             pass
189 # ------------------------------------------------------------------------------------------
190  # |                                    End of Module                                     |
191 # ------------------------------------------------------------------------------------------
192
```

main_menu.py

File Edit View Navigate Code Refactor Run Tools VCS Window Help main_menu.py - main_menu.py

Class 12 Computer Project › main_menu.py ›

1: Project

main_menu.py

```python
# Python Modules
import mysql.connector as conn
import time

# Project Modules
import books
import orders
import acc_ctrl

start_time = time.time()

print(" ----------------------  [ Welcome to S & T Book Shop Management System ]  ----------------------")


def Main_menu(a, b, c):
    # Login in with Account
    mycon = conn.connect(host="localhost", user="root", password="Rinshu@03", database="book_shop")
    login = 0
    while login != 1:  # Checking
        cursor = mycon.cursor()  # for a valid user
        cursor.execute(f"select * from accounts where username = '{a}' and passwd = '{b}' ")
        data = cursor.fetchone()
        if data == None:
            print("Wrong Credentials")
            break
        else:
            if data[0] == 1:   # Checking
                while c != 1:  # for the root(admin) user
                    print("    Successfully Logged In. \n"
                          "    You are Administrator :)")
                    c = 1
                print("Select your choice: \n"
```

7: Structure

2: Favorites

AWS Explorer

```
print("Select your choice: \n"
      "    1 -> Add A New Book \n" +
      "    2 -> Update A book Information \n" +
      "    3 -> Remove A book \n" +
      "    4 -> View Details of a book \n" +
      "    5 -> View All Books \n" +
      "    6 -> Place an Order \n" +
      "    7 -> Cancel an Order \n"
      "    8 -> Return/Replace an Order \n" +
      "    9 -> Update an Order Address \n" +
      "   10 -> View All Orders \n" +
      "   11 -> View Orders of a User(s) \n"
      "   12 -> View Orders by Date \n" +
      "   13 -> Add a New User \n" +
      "   14 -> Update A User's Credentials \n" +
      "   15 -> Remove A User \n"
      "   16 -> View all Users \n"
      "   17 -> View Details of some users \n" +
      "   18 -> Exit")
ch = int(input("Enter your choice from (1 to 18): "))
if ch == 1:
    books.Add_book()
elif ch == 2:
    books.Update_book()
elif ch == 3:
    books.Delete_book()
elif ch == 4:
    books.View_details()
elif ch == 5:
    books.View_all()
elif ch == 6:
    orders.Place_ord(a)
```

```
64          elif ch == 7:
65              orders.Cancel_ord()
66          elif ch == 8:
67              orders.Return_replace(a)
68          elif ch == 9:
69              orders.Update_ord_addr()
70          elif ch == 10:
71              orders.View_all_ords()
72          elif ch == 11:
73              orders.View_all_u()
74          elif ch == 12:
75              orders.View_ord_dt()
76          elif ch == 13:
77              acc_ctrl.Add_new()
78          elif ch == 14:
79              acc_ctrl.Update_user()
80          elif ch == 15:
81              acc_ctrl.Del_user()
82          elif ch == 16:
83              acc_ctrl.All_u()
84          elif ch == 17:
85              acc_ctrl.View_user()
86          elif ch == 18:
87              end_time = time.time()
88              print("Successfully Logged Out. Your active time was ", end_time - start_time, "seconds")
89              exit(0)
90          else:
91              print("Wrong Input")
92          login = 1
93          do_again = input("Go Back to Main Menu? (y -> Yes, n -> No and Logout) ")
94          if do_again in ['y', 'yes', 'Y', 'YES']:
95              Main_menu(a, b, c)
```

```
96          else:
97              end_time = time.time()
98              print("Process Successfully Done. Logged Out")
99              print("Your active time was ", end_time - start_time, "seconds")
100             exit(0)
101     else:
102         print("     Successfully Logged In.")
103         print("    1-> Place an Order \n"
104               "    2-> Cancel an Order \n"
105               "    3-> Return or Replace an Order \n"
106               "    4-> Update an Order Address \n"
107               "    5-> View all your Orders \n" +
108               "    6-> Exit")
109         ch2 = int(input("Enter your choice from (1 to 6): "))
110         if ch2 == 1:
111             orders.Place_ord(a)
112         elif ch2 == 2:
113             orders.Cancel_ord_u(a)
114         elif ch2 == 3:
115             orders.Return_replace(a)
116         elif ch2 == 4:
117             orders.Up_ord_au(a)
118         elif ch2 == 5:
119             orders.View_au(a)
120         elif ch2 == 6:
121             end_time = time.time()
122             print("Successfully Logged Out. Your active time was ", end_time - start_time, "seconds")
123             exit(0)
124         else:
125             print("Wrong Input.")
126         login = 1
127         do_again = input("Go Back to Main Menu? (y -> Yes, n -> No and Logout) ")
```

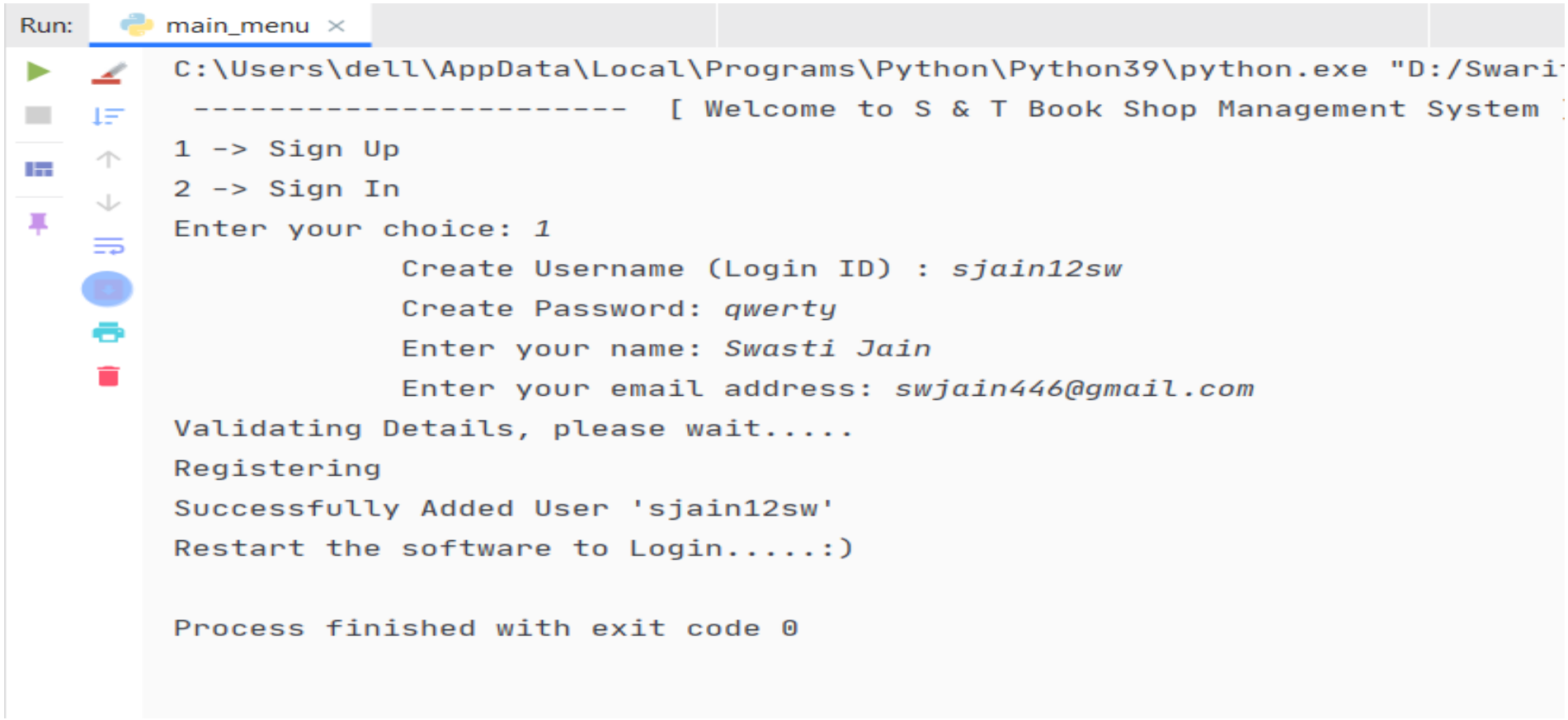
main_menu.py

```
                    if do_again in ['y', 'yes', 'Y', 'YES']:
                        Main_menu(a, b, c)
                    else:
                        end_time = time.time()
                        print("Process Successfully Done. Logged Out")
                        print("Your active time was ", end_time - start_time, "seconds")
                        exit(0)
        log_ag = str(input("Enter login details again? (y -> Yes, n -> No and exit): "))
        if log_ag == 'y' or log_ag == 'Y':
            user_id = Usern()
            user_pass = Userp()
            Main_menu(user_id, user_pass, c)
        else:
            end_time = time.time()
            print("Successfully Logged Out. Your active time was ", end_time - start_time, "seconds")
            exit(0)


#  __ main __
# Calling the main program
print("1 -> Sign Up \n"
      "2 -> Sign In")
sign = int(input("Enter your choice: "))
count = 0

if sign == 1:
    tp = acc_ctrl.Sign_Up()
    print("Restart the software to Login.....:)")
elif sign == 2:

    def Usern():
```

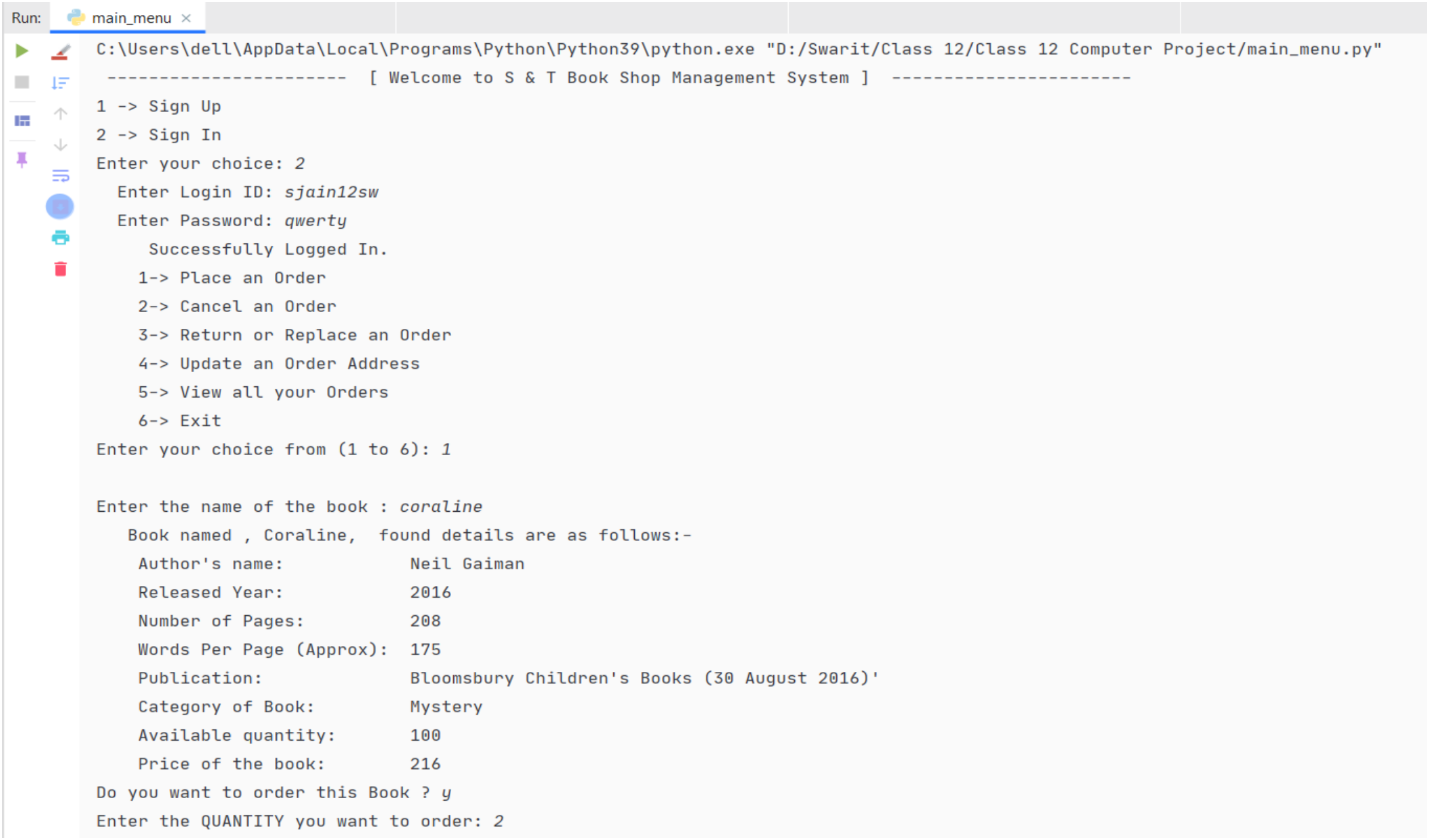
```
158     def Usern():
159         input_username = input("  Enter Login ID: ").strip()
160         return input_username
161
162     def Userp():
163         input_password = input("  Enter Password: ").strip()
164         return input_password
165
166     x = Usern()
167     y = Userp()
168     Main_menu(x, y, count)
169
170 else:
171     print("Wrong Input")
172
```

Outputs of Non-Administrator User

Signing Up


Run:
main_menu
C:\Users\dell\AppData\Local\Programs\Python\Python39\python.exe "D:/Swari
---------- [Welcome to S & T Book Shop Management System]
1 -> Sign Up
2 -> Sign In
Enter your choice: 1
Create Username (Login ID) : sjain12sw
Create Password: qwerty
Enter your name: Swasti Jain
Enter your email address: swjain446@gmail.com
Validating Details, please wait.....
Registering
Successfully Added User 'sjain12sw'
Restart the software to Login.....:)
Process finished with exit code 0

Placing an Order



P.T.O.

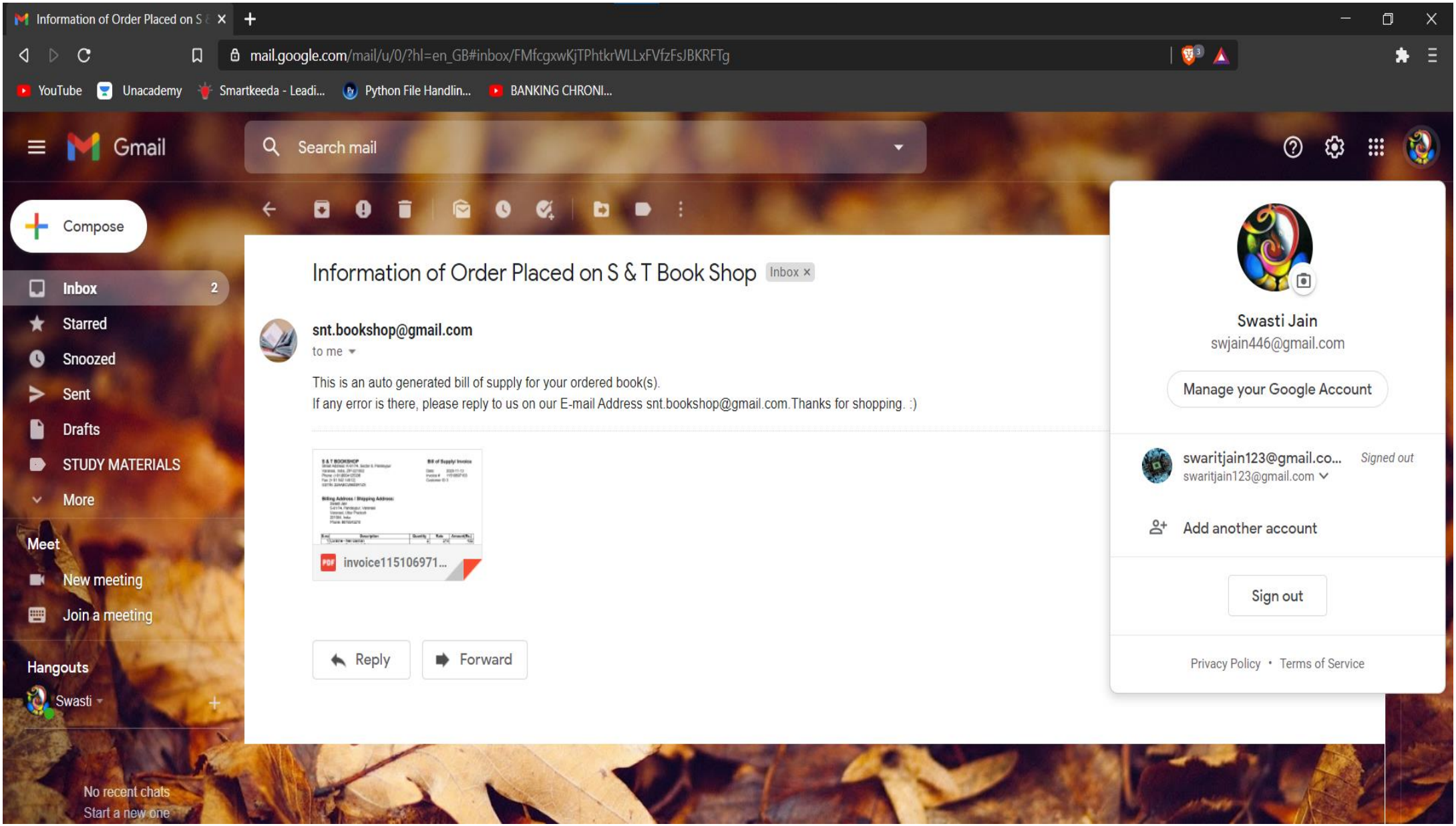
Run: main_menu ×

```
Enter the QUANTITY you want to order: 2
              Checking Stock, please wait...
              Enough quantity found
Please fill up your details below:
     Enter Name: Swasti Jain
     Enter Address: S-61/74, Pandeypur
     Enter District: Varanasi
     Enter City: Varanasi
     Enter State/UT: Uttar Pradesh
     Enter Pincode: 221004
     Enter Country: India
     Enter Phone number: 9876543210
  Now we came at the final step of the process...
             Total Amount to be paid is:  432
             No Delivery Charges Applicable, i.e. Delivery Free
  Are you sure you want to confirm this order? y
Processing your order....

               Order Placed Successfully.

Want to place more orders? (y/n): y

Enter the name of the book : American Gods
   Book named , American Gods,  found details are as follows:-
    Author's name:            Neil Gaiman
    Released Year:            2002
    Number of Pages:          736
    Words Per Page (Approx):  145
```

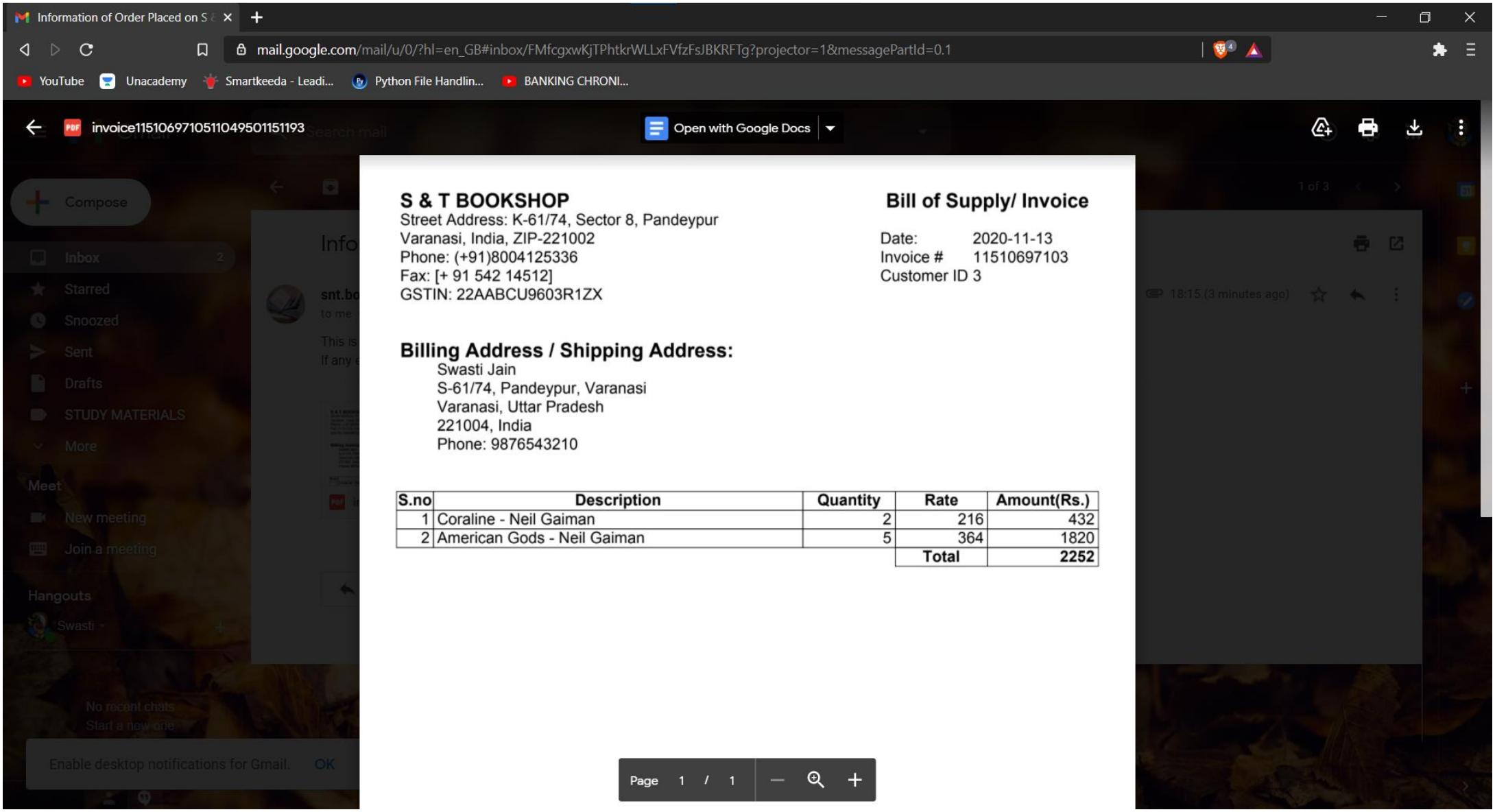
Run: main_menu

```
    Publication:              Headline Review (4 March 2002)
    Category of Book:         Drama
    Available quantity:       12
    Price of the book:        364
Do you want to order this Book ? y
Enter the QUANTITY you want to order: 5
              Checking Stock, please wait...
              Enough quantity found
Please fill up your details below:
     Enter Name: Swasti Jain
     Enter Address: S-61/74, Pandeypur
     Enter District: Varanasi
     Enter City: Varanasi
     Enter State/UT: Uttar Pradesh
     Enter Pincode: 221004
     Enter Country: India
     Enter Phone number: 9876543210
  Now we came at the final step of the process...
            Total Amount to be paid is:  1820
            No Delivery Charges Applicable, i.e. Delivery Free
  Are you sure you want to confirm this order? y
Processing your order....

               Order Placed Successfully.

Want to place more orders? (y/n): n
    Do you want the invoice to be sent to your email address? (y/n): y
An email has been sent to you with an invoice to your email address swjain446@gmail.com
Thanks for visiting us :)

Go Back to Main Menu? (y -> Yes, n -> No and Logout) n
Process Successfully Done. Logged Out
Your active time was  184.12249279022217 seconds

Process finished with exit code 0
```

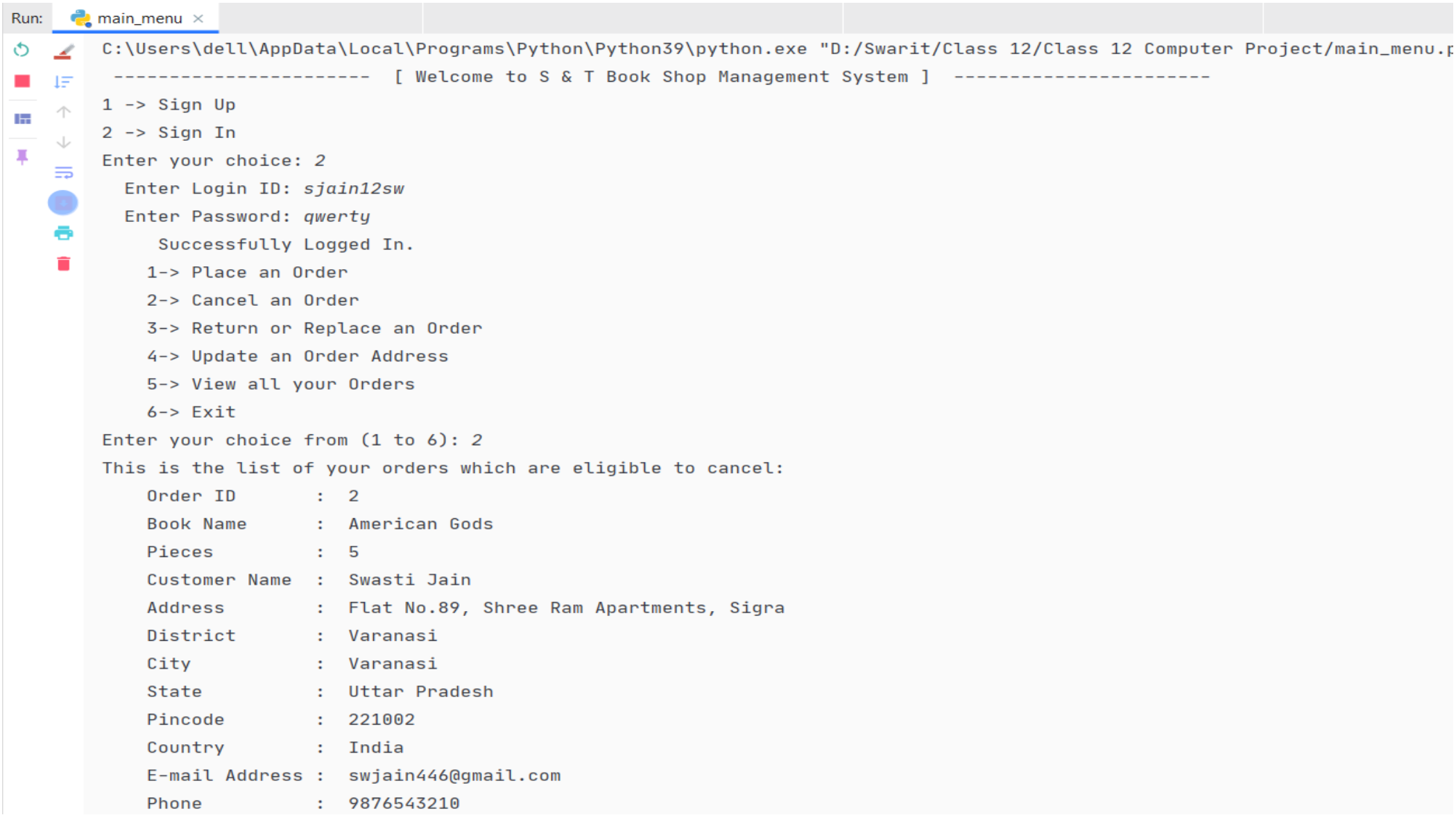
Email Sent on Placing Order on Email When Signing Up



Auto Generated Invoice Sent as PDF to Email Address of the User



Cancelling an Order

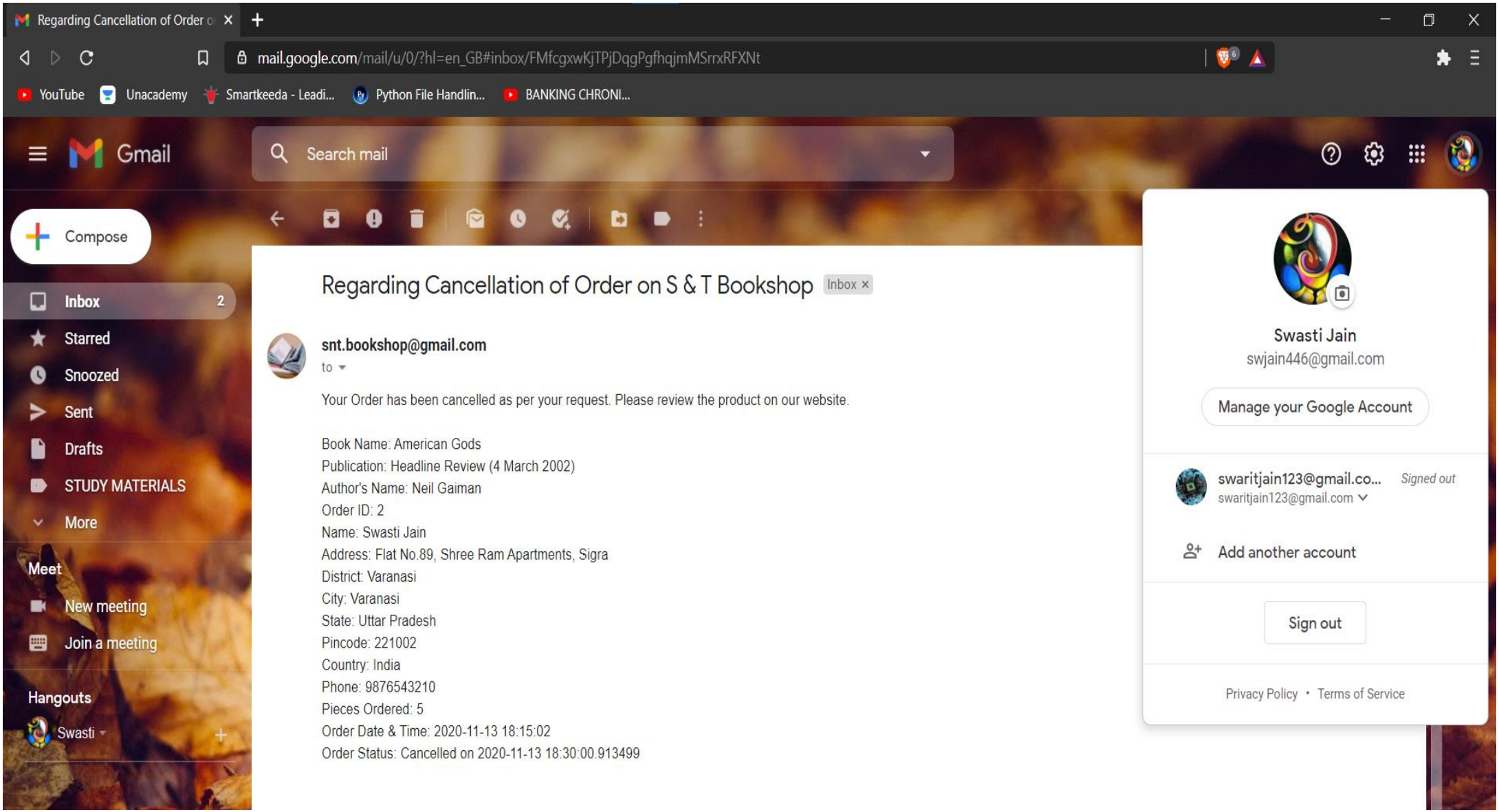


P.T.O

Run: main_menu ×

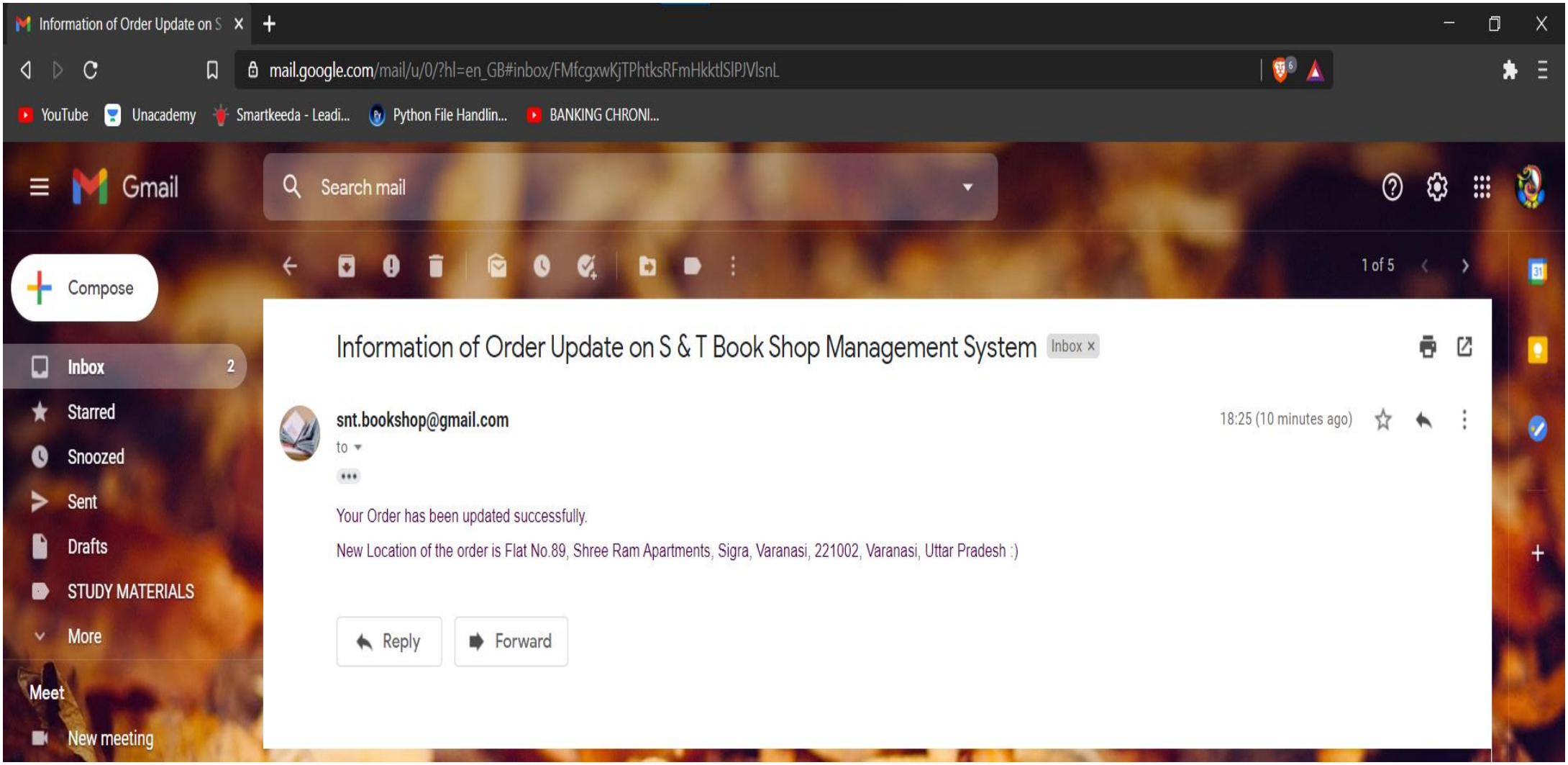
```
    Phone           :  9876543210
    Order Date      :  2020-11-13 18:15:02
    Order Type      :  new
    Order Status    :  In transit

    Order ID        :  1
    Book Name       :  Coraline
    Pieces          :  2
    Customer Name   :  Swasti Jain
    Address         :  S-61/74, Pandeypur
    District        :  Varanasi
    City            :  Varanasi
    State           :  Uttar Pradesh
    Pincode         :  221004
    Country         :  India
    E-mail Address  :  swjain446@gmail.com
    Phone           :  9876543210
    Order Date      :  2020-11-13 18:14:02
    Order Type      :  new
    Order Status    :  In transit

Enter the Order ID to cancel: 2
An email with all details is sent to you on your E-mail Address swjain446@gmail.com
Successfully Cancelled the Order.
   Want to cancel more Orders? (y OR n) n
Go Back to Main Menu? (y -> Yes, n -> No and Logout) y
```

Email Sent on Cancelling the Order



```
Enter your choice from (1 to 6): 4
This is the list of your orders eligible for updating the address:
    Order ID:        1
    Book Name:       Coraline
    Pieces:          2
    Customer Name:   Swasti Jain
    Address:         S-61/74, Pandeypur
    District:        Varanasi
    City:            Varanasi
    State/UT:        Uttar Pradesh
    Pincode:         221004
    Country:         India
    E-mail Address:  swjain446@gmail.com
    Phone:           9876543210
    Order Date:      2020-11-13 18:14:02

Enter the Order ID to update address: 1
Enter the new address: Flat No.89, Shree Ram Apartments, Sigra
Enter City: Varanasi
Enter pincode: 221002
Enter District: Varanasi
Enter State/UT: Uttar Pradesh
Successfully Updated
An email with all details is sent to you on your E-mail Address swjain446@gmail.com

Go Back to Main Menu? (y -> Yes, n -> No and Logout) n
Process Successfully Done. Logged Out
Your active time was  318.53522515296936 seconds
```

Email Sent on Updating Order Address

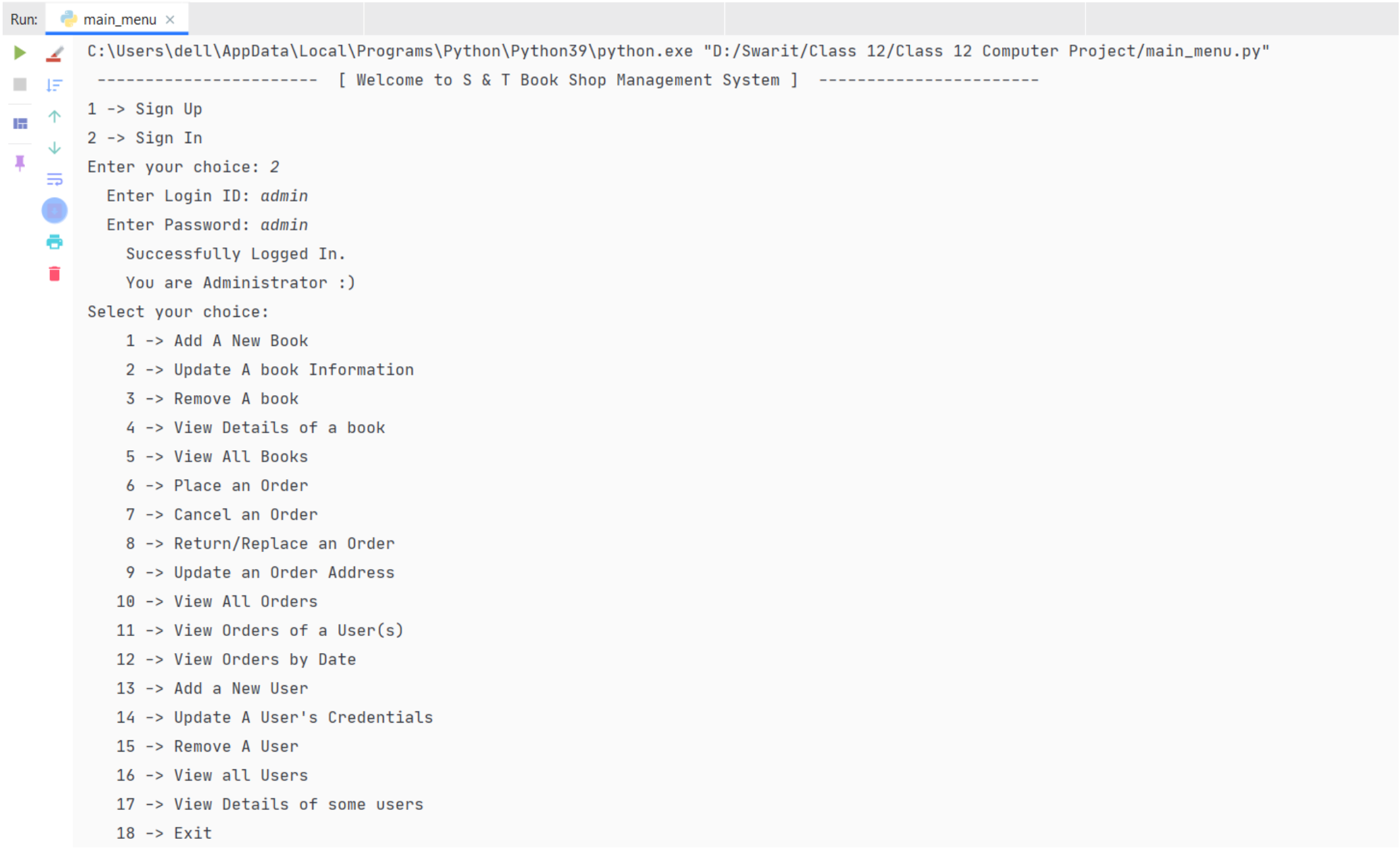


Exiting

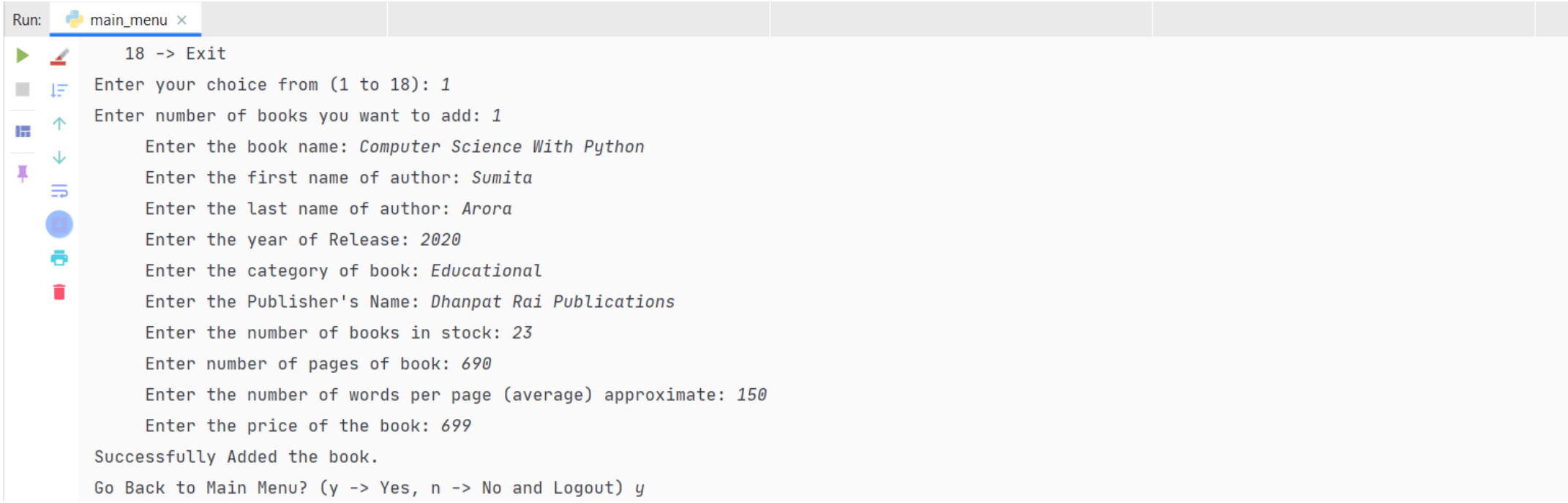
```
Go Back to Main Menu? (y -> Yes, n -> No and Logout) y
     Successfully Logged In.
    1-> Place an Order
    2-> Cancel an Order
    3-> Return or Replace an Order
    4-> Update an Order Address
    5-> View all your Orders
    6-> Exit
Enter your choice from (1 to 6): 6
Successfully Logged Out. Your active time was  66.1079671382904 seconds

Process finished with exit code 0
```

Outputs of Administrator User

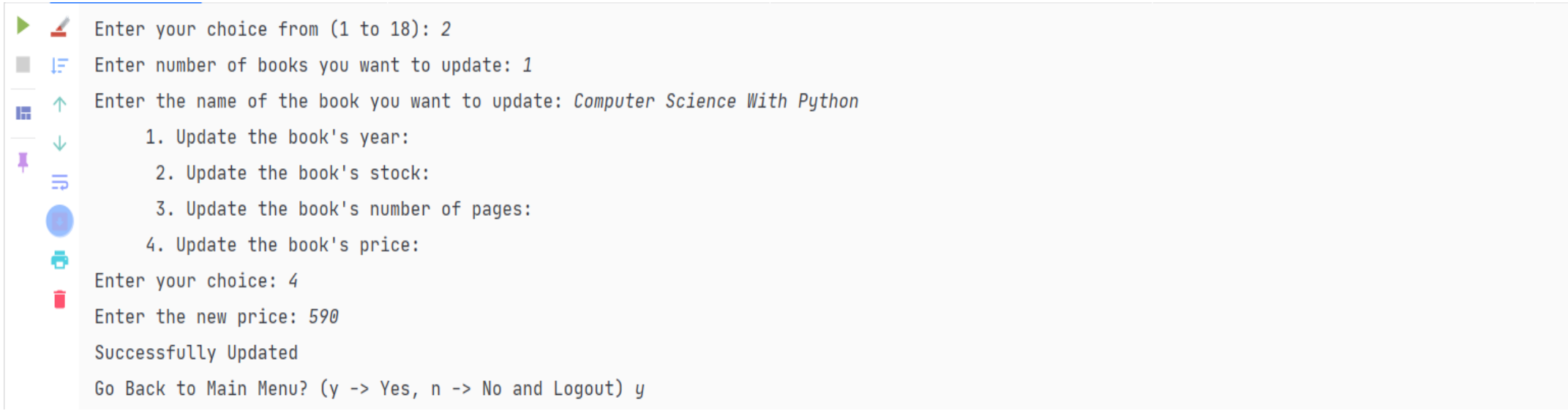
Main Menu of Admin User



Adding a Book



Updating a Book



Deleting a Book

```
Enter your choice from (1 to 18): 3
Enter number of books you want to delete: 1
Enter the name of the book you want to delete: Computer Science With Python
Successfully Deleted.
Go Back to Main Menu? (y -> Yes, n -> No and Logout) n
Process Successfully Done. Logged Out
Your active time was  30.177558183670044 seconds

Process finished with exit code 0
```

Adding Users

```
Enter your choice from (1 to 18): 13
Enter number of user you want to add: 2
    Enter Login ID for the user: swarit_test1
    Enter Password of the user: swarit_test1
    Enter name of the user: Swarit
    Enter email of the user: swaritjain693@gmail.com
Validating Details, please wait.....
Registering
Successfully Added User 'swarit_test1'
    Enter Login ID for the user: swarit_test2
    Enter Password of the user: swarit_test2
    Enter name of the user: Swarit
    Enter email of the user: aayushia234@gmail.com
Validating Details, please wait.....
Registering
Successfully Added User 'swarit_test2'
Go Back to Main Menu? (y -> Yes, n -> No and Logout) n
```

Updating Users

```
Enter your choice from (1 to 18): 14
Enter number of user you want to update: 1
    Enter the username of the user you want to update: swarit_test1
     1 -> Update the user's username
     2 -> Update the user's password
     3 -> Update the user's name
     4 -> Update the user's email address
Enter your choice (1 to 4): 3
    Enter the new name of the swarit_test1: Swarit Jain test1
Successfully Updated
Go Back to Main Menu? (y -> Yes, n -> No and Logout) y
```

Removing User

```
Enter your choice from (1 to 18): 15
Enter number of accounts you want to delete: 2
    Enter the username of the user you want to delete: swarit_test1
Successfully Deleted.
    Enter the username of the user you want to delete: swarit_test2
Successfully Deleted.
Go Back to Main Menu? (y -> Yes, n -> No and Logout) y
```

View All Users

```
Enter your choice from (1 to 18): 16

          User 1
      ID              :  1
      Username        :  admin
      Password        :  admin
      Name            :  admin
      E-mail Address  :  swaritjain123@gmail.com

          User 2
      ID              :  2
      Username        :  testuser
      Password        :  testuser
      Name            :  test
      E-mail Address  :  tanishq.rc@gmail.com

          User 3
      ID              :  3
      Username        :  sjain12sw
      Password        :  qwerty
      Name            :  Swasti Jain
      E-mail Address  :  swjain446@gmail.com
Go Back to Main Menu? (y -> Yes, n -> No and Logout)
```

View Details of a Particular User

```
Enter your choice from (1 to 18): 17
Enter the number of users of which you want details of: 1
    Enter the username of the user: sjain12sw
   ID:            3
   Username:      sjain12sw
   Name:          Swasti Jain
   Email-Address: swjain446@gmail.com

Go Back to Main Menu? (y -> Yes, n -> No and Logout)
```

Returning an Order

Run: main_menu ×

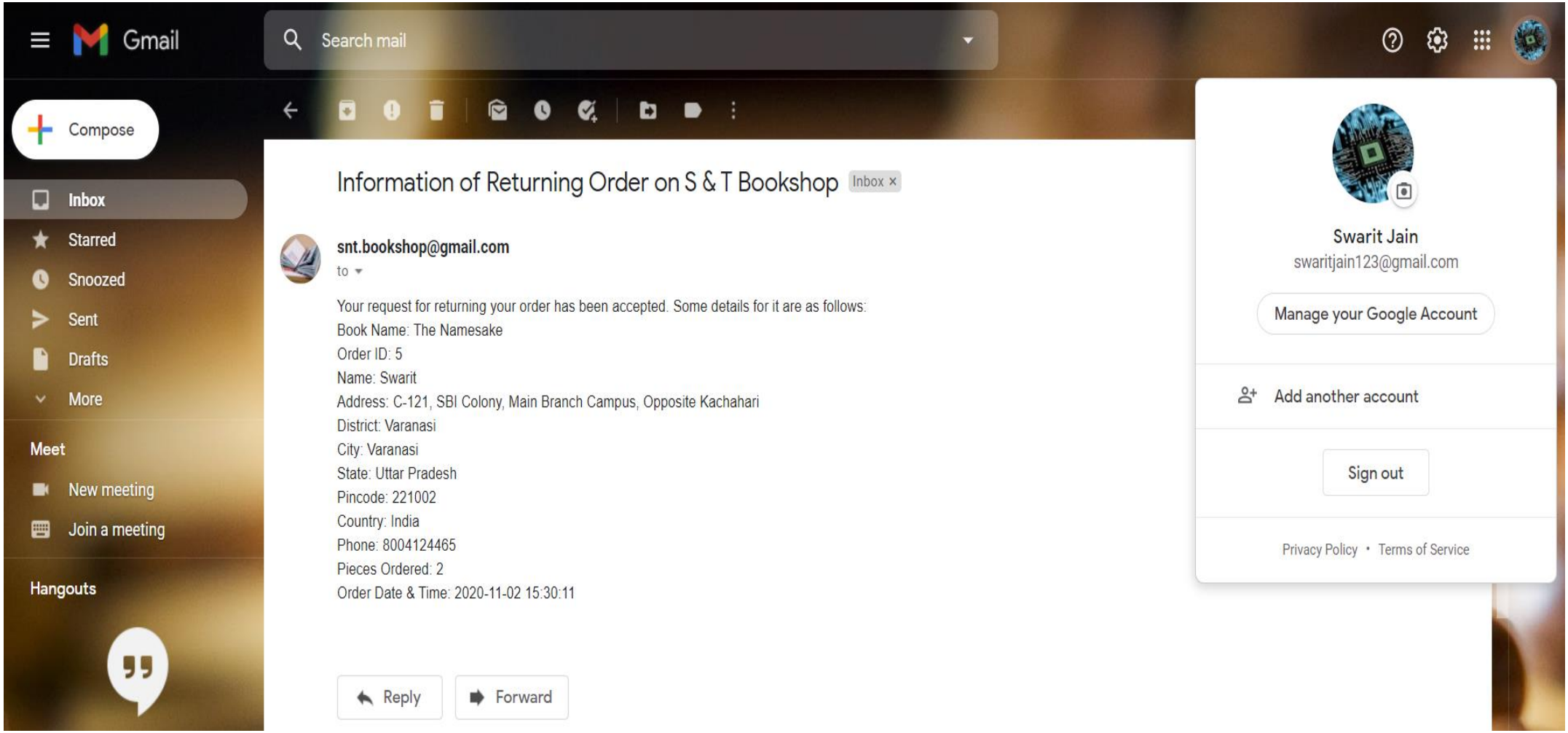
```
Enter your choice from (1 to 18): 8
We have found 2 orders have been delivered to you
List of the orders that have been delivered to you are as follows:-
   Book Name      :  the namesake
   Order Date     :  2020-11-02 15:30:11
   Order ID       :  5
   Customer Name  :  Swarit
   Location       :  C-121, SBI Colony, Main Branch Campus, Opposite Kachahari, Varanasi, Varanasi, Uttar Pradesh, 221002, India
   Email          :  swaritjain123@gmail.com
   Phone no       :  8004124465
   No of Books    :  2

   Book Name      :  coraline
   Order Date     :  2020-11-01 15:30:11
   Order ID       :  4
   Customer Name  :  Swarit
   Location       :  C-121, SBI Colony, Main Branch Campus, Opposite Kachahari, Varanasi, Varanasi, Uttar Pradesh, 221002, India
   Email          :  swaritjain123@gmail.com
   Phone no       :  8004124465
   No of Books    :  1

Do you want to return or replace any of the above orders (y/n) y
Enter the order ID for which you want to return or replace: 5
You have chosen order ID:  5
Few details of your order are as follows:
```

```
   Book name:  the namesake
   Quantity:  2
   Order date:  2020-11-02 15:30:11


By SNT.bookshop Return and Replace policies, you have below two options
1. Return
2. Replace
Enter the option number that you want to select: 1
You have entered under our Return policy
Are you sure you want to return this order? (y/n) y
   Processing the Return
An email with all details is sent to you on your E-mail Address swaritjain123@gmail.com
   Successfully requested for return to your order.
   Our courier boy will come within the next day to collect the package after verifying its condition,    be ready with the packed book
   Money will be refund to your account within 2-3 days
Go Back to Main Menu? (y -> Yes, n -> No and Logout) y
```

Note: - Orders Shown here are test orders of ‘admin’ user which were inserted after the testuser (sjain12sw) placed his/her order. Two Orders displayed above are used one by one for return and replacement.

P.T.O.

Email sent on Returning the order



P.T.O.

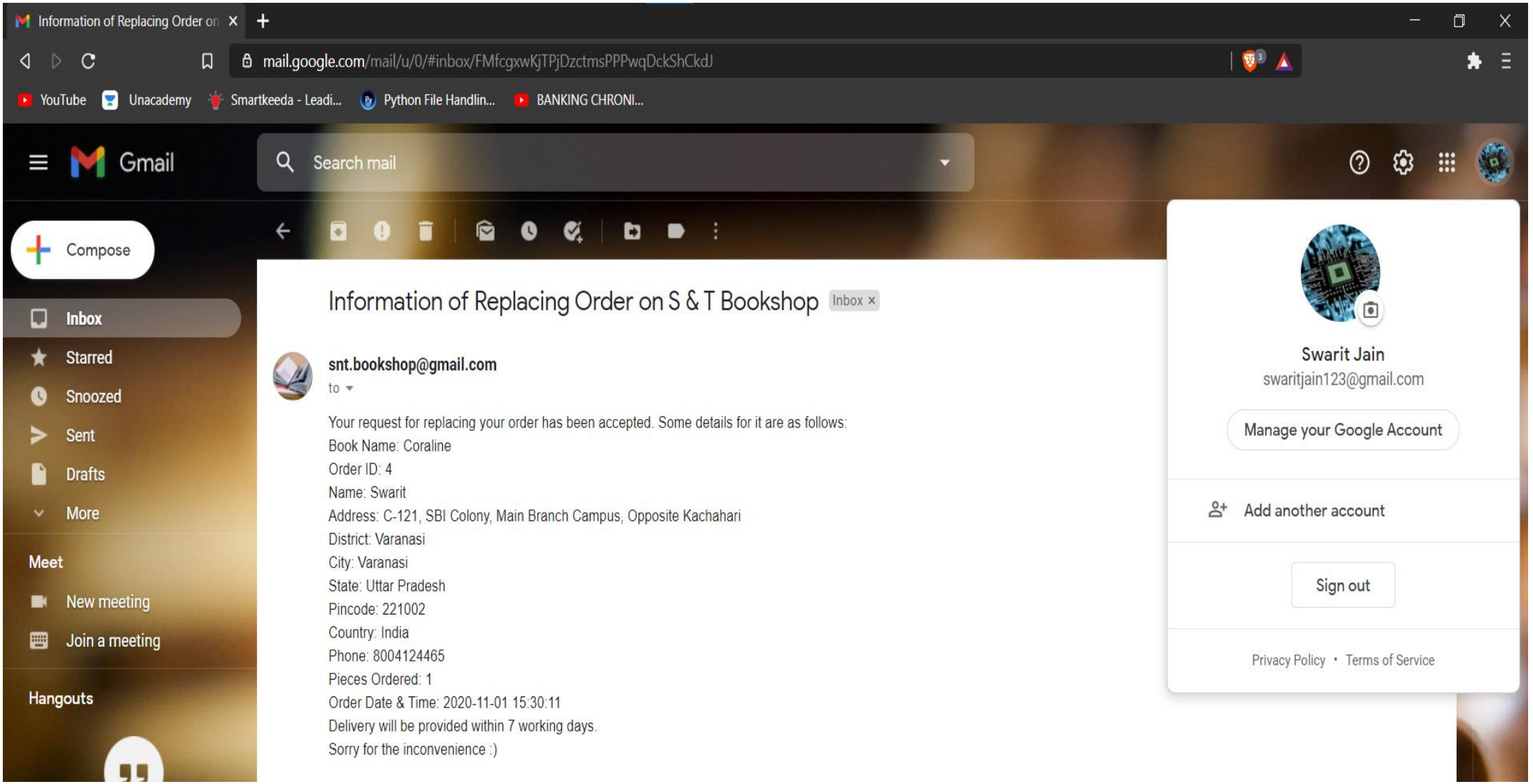
Replacing an Order

Run: main_menu

```
Enter your choice from (1 to 18): 8
We have found 1 orders have been delivered to you
List of the orders that have been delivered to you are as follows:-
   Book Name      :  coraline
   Order Date     :  2020-11-01 15:30:11
   Order ID       :  4
   Customer Name  :  Swarit
   Location       :  C-121, SBI Colony, Main Branch Campus, Opposite Kachahari, Varanasi, Varanasi, Uttar Pradesh, 221002, India
   Email          :  swaritjain123@gmail.com
   Phone no       :  8004124465
   No of Books    :  1

Do you want to return or replace any of the above orders (y/n) y
Enter the order ID for which you want to return or replace: 4
You have chosen order ID:  4
Few details of your order are as follows:

   Book name:  coraline
   Quantity:  1
   Order date:  2020-11-01 15:30:11

By SNT.bookshop Return and Replace policies, you have below two options
1. Return
2. Replace
Enter the option number that you want to select: 2
You have entered under our Replace policy.
Are you sure you want to replace for this order? (y/n) y
   Successfully requested for replacement of your order.
```

```
An email with all details is sent to you on your E-mail Address swaritjain123@gmail.com
   Our courier boy will come within the next day to collect the package after verifying its condition,    be ready with the packed book
   Replacement will be completed within 7 working days until your order has been delivered.
Go Back to Main Menu? (y -> Yes, n -> No and Logout) n
```

Email Sent on Replacing the Order



Viewing all Orders for Replacement

```
Enter your choice from (1 to 18): 10
  1 -> View all New Orders Placed
  2 -> View all Orders for Return
  3 -> View all Orders for Replacement
  4 -> View all Orders including all categories
  Enter your choice (1 to 4): 3
    Order ID       :  8
    Book Name      :  Coraline
    Pieces         :  1
    Customer Name  :  Swarit
    Address        :  C-121, SBI Colony, Main Branch Campus, Opposite Kachahari
    District       :  Varanasi
    City           :  Varanasi
    State          :  Uttar Pradesh
    Pincode        :  221002
    Country        :  India
    E-mail Address :  swaritjain123@gmail.com
    Phone          :  8004124465
    Order Date     :  2020-11-13 20:13:48
    Username       :  admin

Go Back to Main Menu? (y -> Yes, n -> No and Logout)
```

P.T.O.

Viewing all Orders for Return

```
Enter your choice from (1 to 18): 10
  1 -> View all New Orders Placed
  2 -> View all Orders for Return
  3 -> View all Orders for Replacement
  4 -> View all Orders including all categories
  Enter your choice (1 to 4): 2
    Order ID       :  5
    Book Name      :  The Namesake
    Pieces         :  2
    Customer Name  :  Swarit
    Address        :  C-121, SBI Colony, Main Branch Campus, Opposite Kachahari
    District       :  Varanasi
    City           :  Varanasi
    State          :  Uttar Pradesh
    Pincode        :  221002
    Country        :  India
    E-mail Address :  swaritjain123@gmail.com
    Phone          :  8004124465
    Order Date     :  2020-11-02 15:30:11
    Username       :  admin
```

P.T.O.

Viewing all Orders of a particular User (here, admin)

Run: main_menu

```
Enter your choice from (1 to 18): 11
Enter the customer's username: admin
    Order ID       :  8
    Book Name      :  Coraline
    Pieces         :  1
    Customer Name  :  Swarit
    Address        :  C-121, SBI Colony, Main Branch Campus, Opposite Kachahari
    District       :  Varanasi
    City           :  Varanasi
    State          :  Uttar Pradesh
    Pincode        :  221002
    Country        :  India
    E-mail Address :  swaritjain123@gmail.com
    Phone          :  8004124465
    Order Date     :  2020-11-13 20:13:48
    Order Type     :  Replace

    Order ID       :  5
    Book Name      :  The Namesake
    Pieces         :  2
    Customer Name  :  Swarit
    Address        :  C-121, SBI Colony, Main Branch Campus, Opposite Kachahari
    District       :  Varanasi
    City           :  Varanasi
    State          :  Uttar Pradesh
    Pincode        :  221002
    Country        :  India
    E-mail Address :  swaritjain123@gmail.com
    Phone          :  8004124465
    Order Date     :  2020-11-02 15:30:11
    Order Type     :  Return
```

```
Order ID       :  3
Book Name      :  American Gods
Pieces         :  5
Customer Name  :  Swarit
Address        :  C-121, SBI Colony, Main Branch Campus, Opposite Kachahari
District       :  Varanasi
City           :  Varanasi
State          :  Uttar Pradesh
Pincode        :  221002
Country        :  India
E-mail Address :  swaritjain123@gmail.com
Phone          :  8004124465
Order Date     :  2020-10-26 15:30:11
Order Type     :  New
Order Status   :  Delivered on 2020-11-02
```

View all Orders by date

```
Enter your choice from (1 to 18): 12
Enter date (YYYY-MM-DD): 2020-10-26
    Order ID       :  3
    Book Name      :  American Gods
    Pieces         :  5
    Customer Name  :  Swarit
    Address        :  C-121, SBI Colony, Main Branch Campus, Opposite Kachahari
    District       :  Varanasi
    City           :  Varanasi
    State          :  Uttar Pradesh
    Pincode        :  221002
    Country        :  India
    E-mail Address :  swaritjain123@gmail.com
    Phone          :  8004124465
    Order Date     :  2020-10-26 15:30:11
    Username       :  admin
    Order Type     :  New
    Order Status   :  Delivered on 2020-11-02

Go Back to Main Menu? (y -> Yes, n -> No and Logout) n
```

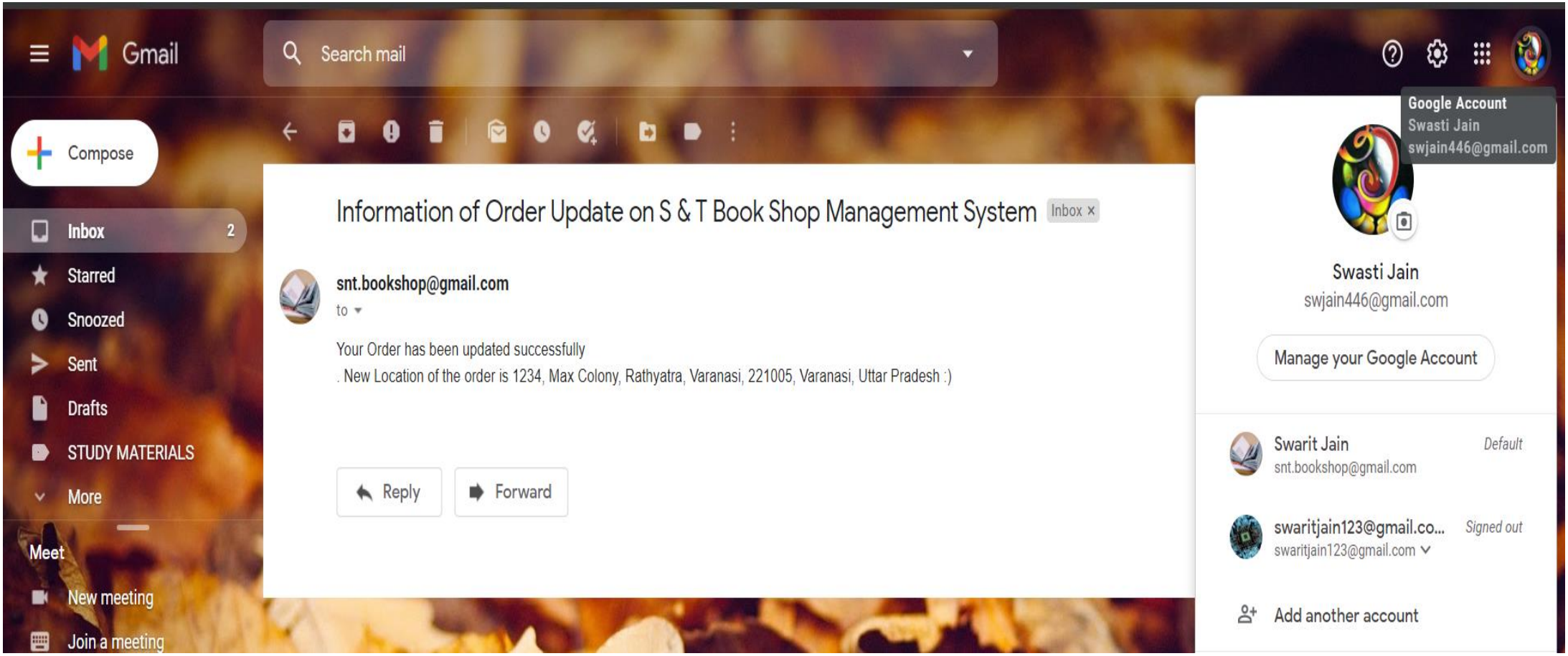
Updating an Order Address

```
Enter your choice from (1 to 18): 9
Only 2 order(s) is/are eligible for update of address.
Enter the number of orders's addresses you want to update: 1
Enter the Order ID to update the order: 1
     Enter the new address for order ID 1: 1234, Max Colony, Rathyatra
     Enter the new city for order ID    1: Varanasi
     Enter the new pincode for order ID 1: 221005
     Enter the new district for Order ID1: Varanasi
     Enter the new State/UT for Order ID1: Uttar Pradesh
An email with all details is sent to you on your E-mail Address swjain446@gmail.com
Successfully Updated
regarding order update for new location on email address swjain446@gmail.com
Go Back to Main Menu? (y -> Yes, n -> No and Logout)
```

Note: - Here Order ID ‘1’ is of the testuser(sjain12sw).

P.T.O

Email Sent on Order Address being Updated by Admin



P.T.O.

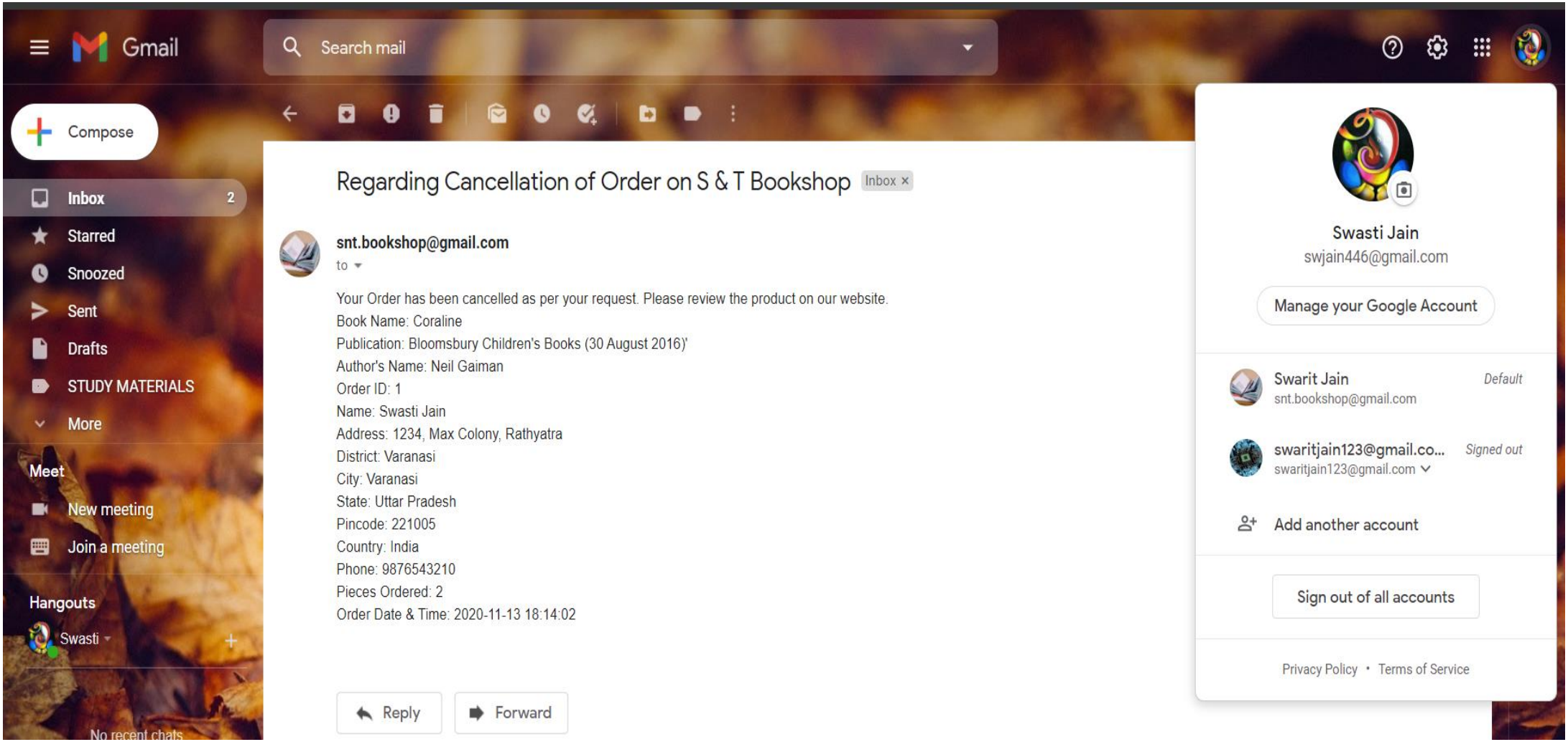
Cancelling an Order

```
Enter your choice from (1 to 18): 7
Only 2 order(s) is/are eligible to cancel.
Enter number of orders you want to cancel (not more than 2): 1
  Enter the Order ID: 1
An email with all details is sent to you on your E-mail Address swjain446@gmail.com
Successfully cancelled your order.
Go Back to Main Menu? (y -> Yes, n -> No and Logout) n
```

Note: - Here, order is being cancelled by admin and email is sent to the user's email address on which the order was placed.

P.T.O.

Email Sent on Order being Cancelled by Admin



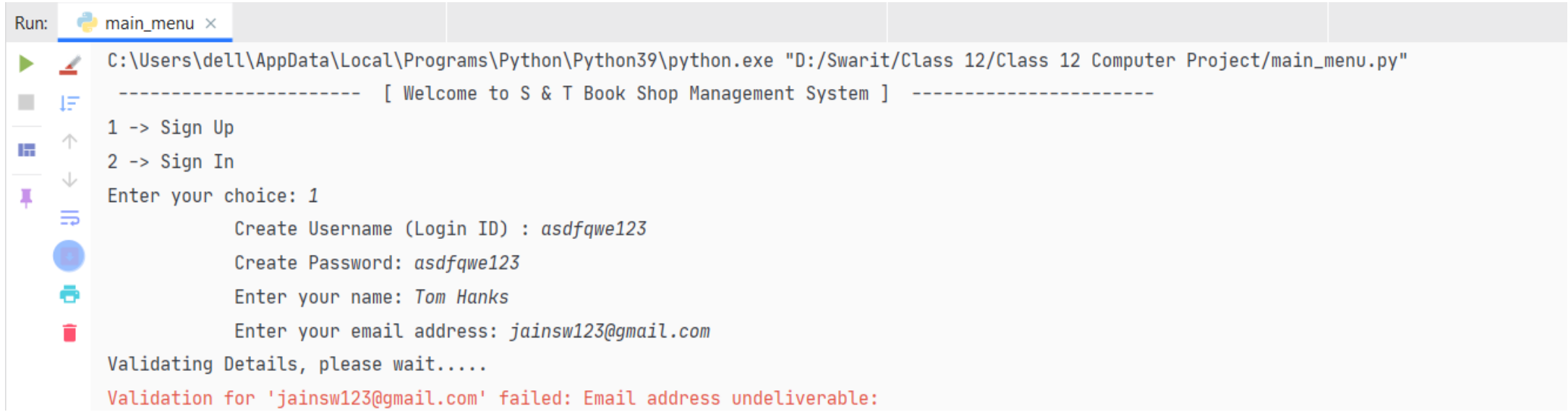
Exiting from Administrator Block

```
Enter your choice from (1 to 18): 18

Successfully Logged Out. Your active time was  27.018373727798462 seconds
```

Error Handling

Already Registered Email Address can’t be used by any other user.

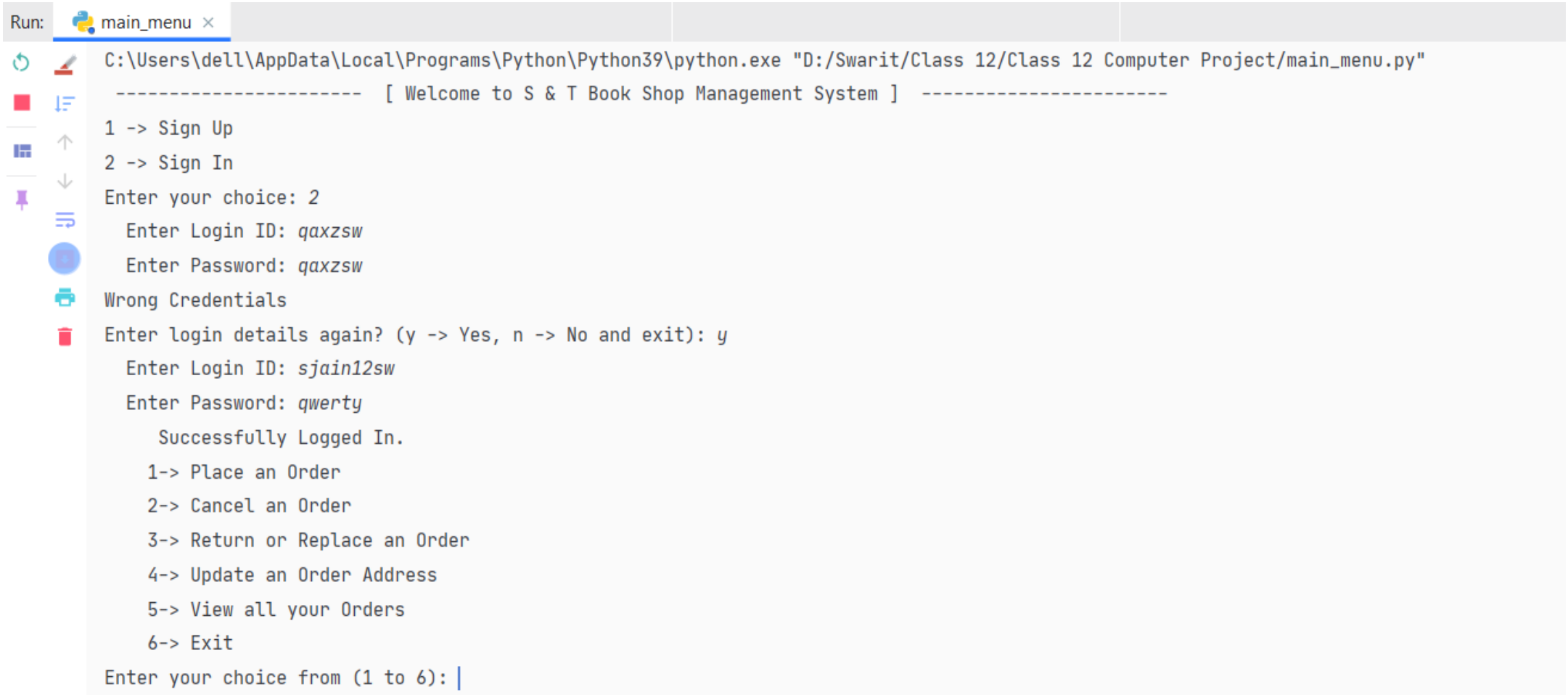
Email Validation (can’t insert invalid email(s))



Admin account can’t be deleted.

```
   13 -> Add a New User
   14 -> Update A User's Credentials
   15 -> Remove A User
   16 -> View all Users
   17 -> View Details of some users
   18 -> Exit
Enter your choice from (1 to 18): 15
Enter number of accounts you want to delete: 1
    Enter the username of the user you want to delete: admin
Can't Delete user. The specified username is Administrator.
Go Back to Main Menu? (y -> Yes, n -> No and Logout)
```

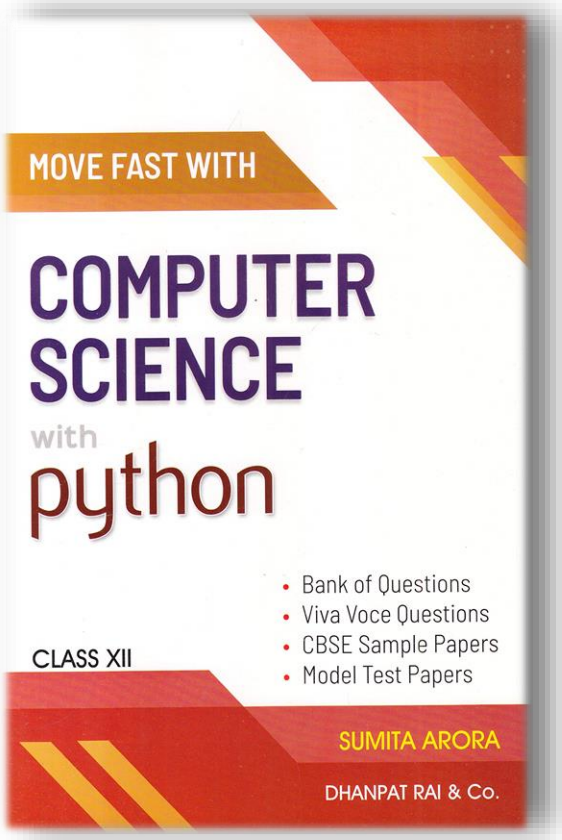
Can’t login by invalid credentials



Bibliography

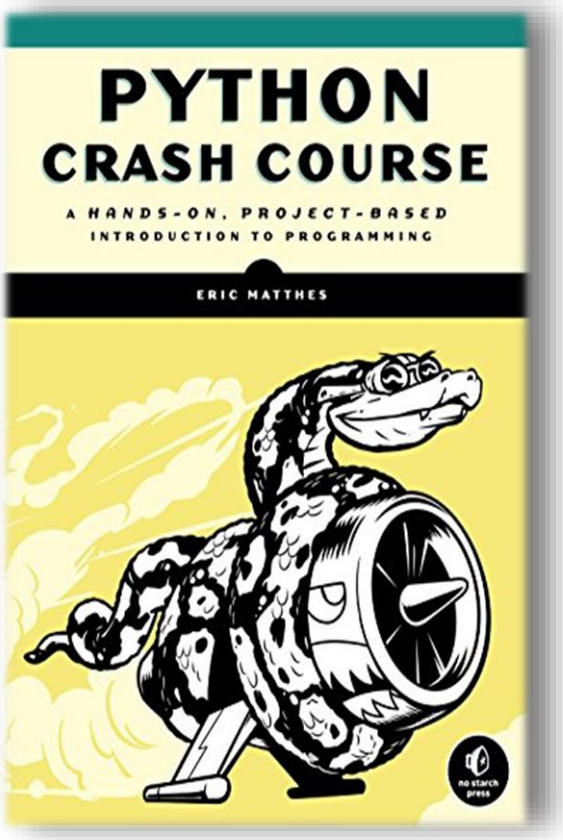
References

1. https://www.w3schools.com
2. https://www.wikipedia.org
3. https://www.stackoverflow.com
4. https://www.geekforgeeks.org



Computer Science with Python – Sumita Arora

Class 12



Python Crash Course – Eric Matthes