**Project: Application for Library**

Navya Sree Guduri

MSITM, University of Texas at Dallas

MIS-6382: Object Oriented Program Python

Prof. Vastal Kamleshbhai Maru

December 13, 2022

**Application for Library**

Aiming to create an application for library where a customer can rent a book for an hour OR a day OR a week. Hourly price for renting a book is $1, daily price for renting a book is $2, and weekly price for renting a book is $5. A person can rent more than one book. A library can issue a receipt when customers return a book. Total due is shown by calculating respective rates and time. A customer can select more books but not different time periods and that the request cannot exceed the number of books available.

**Flow Chart:**

Diagram

Description automatically generated

**Method:**

class Library:

Initiated a class ‘Library’ and passing a parameter number of books available in the shop to rent

def \_\_init\_\_(self, num\_books):

self.num\_books = num\_books

class Customer:

Initiated a class ‘Customer’ and passing parameters number of books and time to choose by customer and initially the total amount due will be zero

def \_\_init\_\_(self, num\_books, time):

self.num\_books = num\_books

self.time = time

self.total\_due = 0

def calculate\_price(self):

Declared the rates to calculate price

If Customer choose hourly or weekly or a day and number of books where it should be more than one and calculate total amount due based on rate

hourly\_rate = 1

daily\_rate = 2

weekly\_rate = 5

if self.time == "hour" and self.num\_books > 1:

total\_due = self.num\_books \* hourly\_rate

return total\_due

elif self.time == "day" and self.num\_books > 1:

total\_due = self.num\_books \* daily\_rate

return total\_due

elif self.time == "week" and self.num\_books > 1:

total\_due = self.num\_books \* weekly\_rate

return total\_due

else:

This function is to check whether customer selected books less than or equal to number of books available to rent and generate invoice with total due and total time otherwise returns message “ Number of books requested exceeds the number of books available”.)

return 0

def issue\_invoice(self, library):

if self.num\_books <= library.num\_books:

self.total\_due = self.calculate\_price()

print(f"Total due: {self.total\_due}")

print(f"Total time: {self.time}")

else:

print("Number of books requested exceeds the number of books available.")

Unit test:

Testing the above code passing parameter value to num\_books as 10 (i.e; number of books available to rent a customer) and passing 5(num of books required by the customer) and its on daily basis.

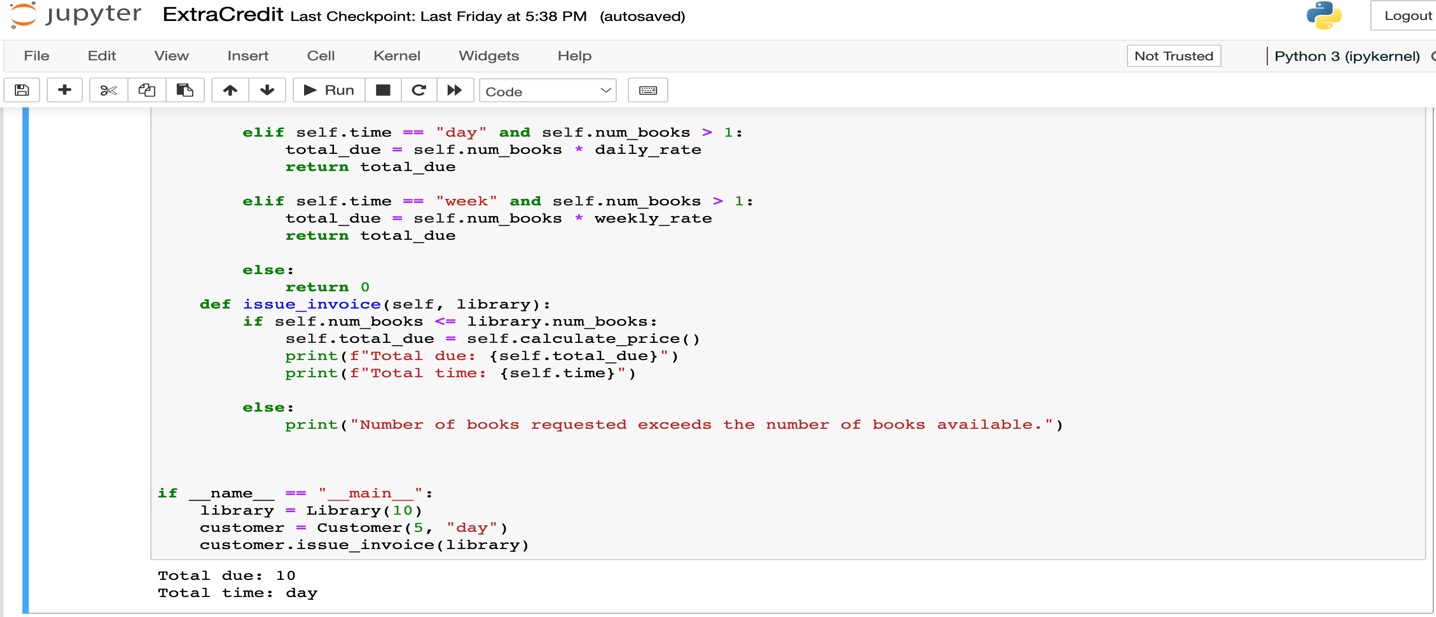
if \_\_name\_\_ == "\_\_main\_\_":

library = Library(10)

customer = Customer(5, "day")

customer.issue\_invoice(library)

**Results:**

****

The customer rents 5 books for a day, which is the equivalent of 5 x 2 = 10.

Since the customer rents more than one book and less than available number of books which is 10, resulting in a total due of 10.

Graphical user interface, text, application, email

Description automatically generated

The customer rents 11 books for a day, which exceeded the limit number of books available to rent. Since the customer rents beyond the limit. The output is a message “ Number of books requested exceeds the number of books available. Thus, no invoice generated.