DELHI PUBLIC SCHOOL BHARUCH

INFORMATICS PRACTICES

INVESTIGATORY PROJECT

2020-21

PYTHON PROJECT:

BOOK STORE MANAGEMENT

NAME: NIDHI MODI,

HEEBA KAYSER ANSARI,

SANSKRITI GUHA

CLASS: XII-A, XII-B

ROLL NO.:

ACKNOWLEDGEMENT

We Have taken efforts in this project. However it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

We also thank respected principal sir for his immense support and motivation regarding this project. We are highly indebted to MEHUL KUMAR DALWADI SIR for his guidance and constant supervision as well as providing necessary information regarding the project. We would like to express our gratitude towards our parents for their cooperation of this project.

Our thanks and appreciation also goes for our friends in developing the project and the people who have willingly helped us out with their abilities.

DELHI PUBLIC SCHOOL

CERTIFICATE

This is to certify that Ms. Sanskriti Guha, Ms. Nidhi Modi and Ms. Heeba Kayser Ansari of class XII has successfully completed the IP Investigatory Project titled 'BOOK STORE MANAGEMENT' during the academic session 2020-21 towards the partial fulfillment of credit for I.P. practical evaluation of class XII CBSE Board Examination and submitted satisfactory report as compiled in the following pages. This report is the result of his/her endeavor and research. It is finalized under my guidance and supervision.

Signature of principal

examiner

OBJECTIVE

The purpose of this project is to let the student apply python programming knowledge to a real-world situation/problem.

The aim of the project is to enhance skills in a student regarding procedural analysis of planning, creating and implementing a programmed record or software to help reduce useless waste of time in recording files in a not-so-favorable manner.

THEORY

{1} python

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation.

{2} pandas

Pandas aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis/ manipulation tool available in any language.

A world where data analytics and manipulation software is:

- Accessible to everyone
- Free for users to use and modify
- Flexible
- Powerful
- Easy to use
- Fast

{3} Series

Series is a one-dimensional labelled array capable of holding data of any type (integer, string, float, **python** objects, etc.). The axis labels are collectively called index.

{4} Dictionary

Dictionaries are **Python's** implementation of a data structure that is more generally known as an associative array. A **dictionary** consists of a collection of key-value pairs. Each key-value pair maps the key to its associated value.

{5} Dataframe

Dataframe is a 2-dimensional labelled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table, or a dict of Series objects. It is generally the most commonly used **pandas** object.

TKINTER

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets. There are currently 15 types of widgets in Tkinter.

CSV FILES

A Comma Separated Values (CSV) file is a plain text file that contains a list of data. These files are often used for exchanging data between different applications. For example, databases and contact managers often support CSV files.

These files may sometimes be called Character Separated Values or Comma Delimited files. They mostly use the comma character to separate (or delimit) data, but sometimes use other characters, like semicolons. The idea is that you can export complex data from one application to a CSV file, and then import the data in that CSV file into another application.

A CSV file has a fairly simple structure. It's a list of data separated by commas.

IDENTIFY PROBLEMS/GET FEEDBACK

Companies can bring in feedback from the customers, stakeholders, to identify the problem, so that it can be fixed smoothly without any hassle. It helps to identify strength and weakness.

- Identify and validate an opportunity to improve business accomplishments of the organization or a deficiency related to a business need.
- Identify significant assumptions and constraints on solutions to that need.

- Recommend the exploration of alternative concepts and methods to satisfy the need including questioning the need for technology, i.e., will a change in the business process offer a solution?
- Assure executive business and executive technical sponsorship. The Sponsor designates a Project Manager and the business need is documented in a Concept Proposal. The Concept Proposal includes information about the business process and the relationship to the Agency/Organization.
- Infrastructure and the Strategic Plan. A successful Concept Proposal results in a Project Management Charter which outlines the authority of the project manager to begin the project.

Requirement Analysis

This phase formally defines the detailed functional user requirements using high-level requirements identified in the Initiation, System Concept, and Planning phases. It also delineates the requirements in terms of data, system performance, security, and maintainability requirements for the system. The requirements are defined in this phase to a level of detail sufficient for systems design to proceed. They need to be measurable, testable, and relate to the business need or opportunity identified in the Initiation Phase. The requirements that will be used to determine acceptance of the system are captured in the Test and Evaluation MasterPlan.

The purposes of this phase are to:

- Further define and refine the functional and data requirements and document them in the Requirements Document,
- Complete business process reengineering of the functions to be supported (i.e., verify what information drives the business process, what information is generated, who generates it, where does the information go, and who processes it),
- Develop detailed data and process models (system inputs, outputs, and the process.
- Develop the test and evaluation requirements that will be used to determine acceptable system performance.

Plan

After determining issues and problems related to the software the next step is 'plan' in the sense how to plan to overcome the problems and fix it in the minimum time and cost as possible. They work out procedures, planning, cost planning, etc. In short the staff should come up with the most effective, and feasible plan.

Design

After planning the next step is making it into a design plan. It is made and can be reviewed and made certain changes so that the cost stays as minimum and effective as possible. Failure at this stage will almost certainly result in cost overruns at best and the total collapse of the project at worst.

- Identifying potential risks and defining mitigating design features.
- Performing a security risk assessment.
- Developing a conversion plan to migrate current data to the new system.
- Determining the operating environment.
- Defining major subsystems and their inputs and outputs.
- Allocating processes to resources.
- Preparing detailed logic specifications for each software module. The result is a draft System Design Document which captures the preliminary design for the system.
- Everything requiring user input or approval is documented and reviewed by the user. Once these documents have been approved by the Agency CIO and Business Sponsor, the final System Design Document is created to serve as the Critical/Detailed Design for the system.
- This document receives a rigorous review by Agency technical and functional representatives to ensure that it satisfies the business requirements. Concurrent with the

development of the system design, the Agency Project Manager begins development of the Implementation Plan, Operations and Maintenance Manual, and the Training Plan.

-

Build

At this stage, the actual development starts. It's important that every developer sticks to the agreed blueprint. Also, make sure you have proper guidelines in place about the code style and practices. This will help to produce organized and consistent code that is easier to understand but also to test during the next phase.

- Translating the detailed requirements and design into system components.
- Testing individual elements (units) for usability.
- Preparing for integration and testing of the IT system.

Code test

In this stage, we test for defects and deficiencies. We fix those issues until the product meets the original specifications.

In short, we want to verify if the code meets the defined requirements.

- Subsystem integration, system, security, and user acceptance testing is conducted during the integration and test phase. The user, with those responsible for quality assurance, validates that the functional requirements, as defined in the functional requirements document, are satisfied by the developed or modified system. OIT Security staff assesses the system security and issue a security certification and accreditation prior to installation/implementation.

Multiple levels of testing are performed, including:

- Testing at the development facility by the contractor and possibly supported by end users

- Testing as a deployed system with end users working together with contract personnel

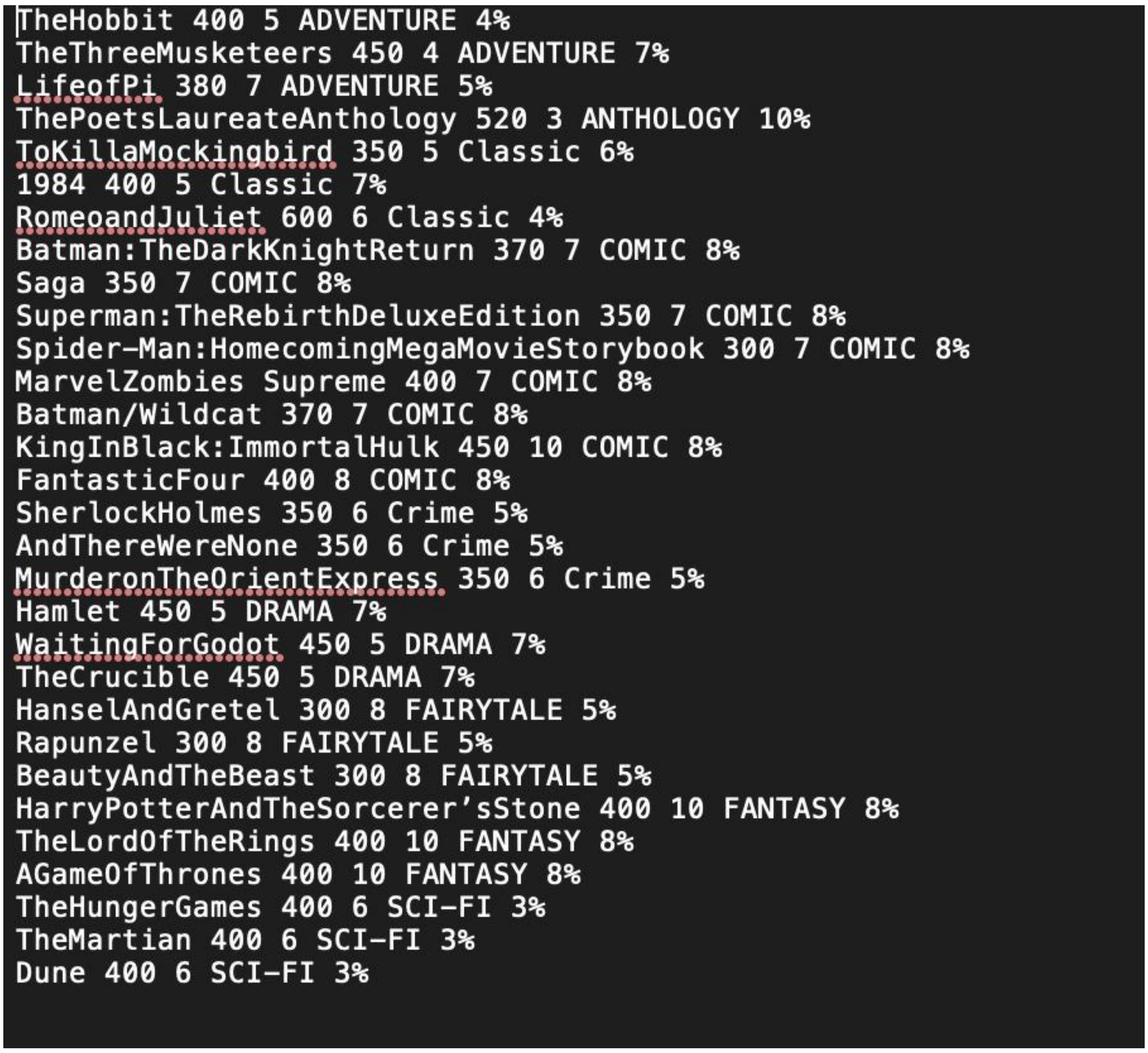
- Operational testing by the end user alone performing all functions. Requirements are traced throughout testing, a final Independent Verification & Validation evaluation is performed and all documentation is reviewed and accepted prior to acceptance of the system.

SOURCE CODE

```
from tkinter import *
from tkinter import messagebox
```

```
import os
f=open("database_proj",'a+')
root = Tk()
root.title("Book Store Managment System")
root.configure(width=1500,height=600,bg='BLACK')
var=-1

def additem():
    global var
    num_lines = 0
    with open("database_proj", 'r') as f10:
        for line in f10:
            num_lines += 1
    var=num_lines-1
    e1= entry1.get()
    e2=entry2.get()
    e3=entry3.get()
    e4=entry4.get()
    e5=entry5.get()
    f.write('{0} {1} {2} {3} {4}\n'.format(str(e1),e2,e3,str(e4),e5))
    entry1.delete(0, END)
    entry2.delete(0, END)
    entry3.delete(0, END)
    entry4.delete(0, END)
    entry5.delete(0, END)



def firstitem():
    global var
    var=0
    f.seek(var)
    c=f.readline()
    v=list(c.split(" "))
    entry1.delete(0, END)
    entry2.delete(0, END)
    entry3.delete(0, END)
    entry4.delete(0, END)
    entry5.delete(0, END)
    entry1.insert(0,str(v[0]))
    entry2.insert(0,str(v[1]))
    entry3.insert(0,str(v[2]))
    entry4.insert(0,str(v[3]))
    entry5.insert(0,str(v[4]))

def nextitem():
    global var
    entry4.delete(0, END)
    entry5.delete(0, END)




def firstitem():
    global var
    var=0
    f.seek(var)
    c=f.readline()
    v=list(c.split(" "))
    entry1.delete(0, END)
    entry2.delete(0, END)
    entry3.delete(0, END)
    entry4.delete(0, END)
    entry5.delete(0, END)
    entry1.insert(0,str(v[0]))
    entry2.insert(0,str(v[1]))
    entry3.insert(0,str(v[2]))
    entry4.insert(0,str(v[3]))
    entry5.insert(0,str(v[4]))

def nextitem():
    global var
        try:
            z = f.readlines()
            xyz=z[var]
            v = list(xyz.split(" "))
            entry1.delete(0, END)
            entry2.delete(0, END)
            entry3.delete(0, END)
            entry4.delete(0, END)
            entry5.delete(0, END)

            entry1.insert(0, str(v[0]))
            entry2.insert(0, str(v[1]))
            entry3.insert(0, str(v[2]))
            entry4.insert(0, str(v[3]))
            entry5.insert(0, str(v[4]))
        except:
            messagebox.showinfo("Title", "SORRY!...NO MORE RECORDS")


def lastitem():
    global var
    f4=open("database_proj",'r')
    x=f4.read().splitlines()
    last_line= x[-1]
    num_lines = 0
    with open("database_proj", 'r') as f8:
        for line in f8:
            num_lines += 1
    var=num_lines-1
    print(last_line)
    try:
        v = list(last_line.split(" "))
        entry1.delete(0, END)
        entry2.delete(0, END)
        entry3.delete(0, END)
        entry4.delete(0, END)
        entry5.delete(0, END)

        entry1.insert(0, str(v[0]))
        entry2.insert(0, str(v[1]))
        entry3.insert(0, str(v[2]))
        entry4.insert(0, str(v[3]))
        entry5.insert(0, str(v[4]))
    except:
        messagebox.showinfo("Title", "SORRY!...NO MORE RECORDS")


def updateitem():

    e1 = entry1.get()
    e2 = entry2.get()
    e3 = entry3.get()
    e4 = entry4.get()
    e5 = entry5.get()
    with open(r"database_proj") as f1, open(r"database_proj1", "w") as working:
        for line in f1:
            if str(e1) not in line:
                working.write(line)
            else:
                working.write('{0} {1} {2} {3} {4}'.format(str(e1), e2, e3, str(e4), e5))
    os.remove(r"database_proj")
    #brought to you by code-projects.org
    os.rename(r"database_proj1", r"database_proj")


def searchitem():
    i=0
    e11 = entry1.get()
    with open(r"database_proj") as working:
        for line in working:
            i=i+1
            if str(e11) in line:
                break
        try:
            v = list(line.split(" "))
            entry1.delete(0, END)
            entry2.delete(0, END)
            entry3.delete(0, END)
            entry4.delete(0, END)
            entry5.delete(0, END)
            entry1.insert(0, str(v[0]))
            entry2.insert(0, str(v[1]))
            entry3.insert(0, str(v[2]))
            entry4.insert(0, str(v[3]))
            entry5.insert(0, str(v[4]))
        except:
            messagebox.showinfo("Title", "error end of file")
    working.close()


def clearitem():
    entry1.delete(0, END)
    entry2.delete(0, END)
    entry3.delete(0, END)
    entry4.delete(0, END)
    entry5.delete(0, END)
#fn1353
label0= Label(root,text="BOOK STORE MANAGEMENT SYSTEM
",bg="darkcyan",fg="white",font=("times", 34))
label1=Label(root,text="ENTER BOOK
NAME",bg="darkcyan",relief="ridge",fg="white",font=("Times", 12),width=25)
entry1=Entry(root , font=("Times", 12))
label2=Label(root, text="ENTER BOOK
PRICE",bd="2",relief="ridge",height="1",bg="darkcyan",fg="white", font=("Times", 12),width=25)
entry2= Entry(root, font=("Times", 12))
label3=Label(root, text="ENTER BOOK
QUANTITY",bd="2",relief="ridge",bg="darkcyan",fg="white", font=("Times", 12),width=25)
entry3= Entry(root, font=("Times", 12))
label4=Label(root, text="ENTER BOOK
CATEGORY",bd="2",relief="ridge",bg="darkcyan",fg="white", font=("Times", 12),width=25)
entry4= Entry(root, font=("Times", 12))
label5=Label(root, text="ENTER BOOK DISCOUNT",bg="darkcyan",relief="ridge",fg="white",
font=("Times", 12),width=25)
entry5= Entry(root, font=("Times", 12))
button1= Button(root, text="ADD BOOK", bg="white", fg="black", width=20, font=("Times",
12),command=additem)
button2= Button(root, text="VIEW FIRST BOOK" , bg="white", fg="black", width =20,
font=("Times", 12),command=firstitem)
```

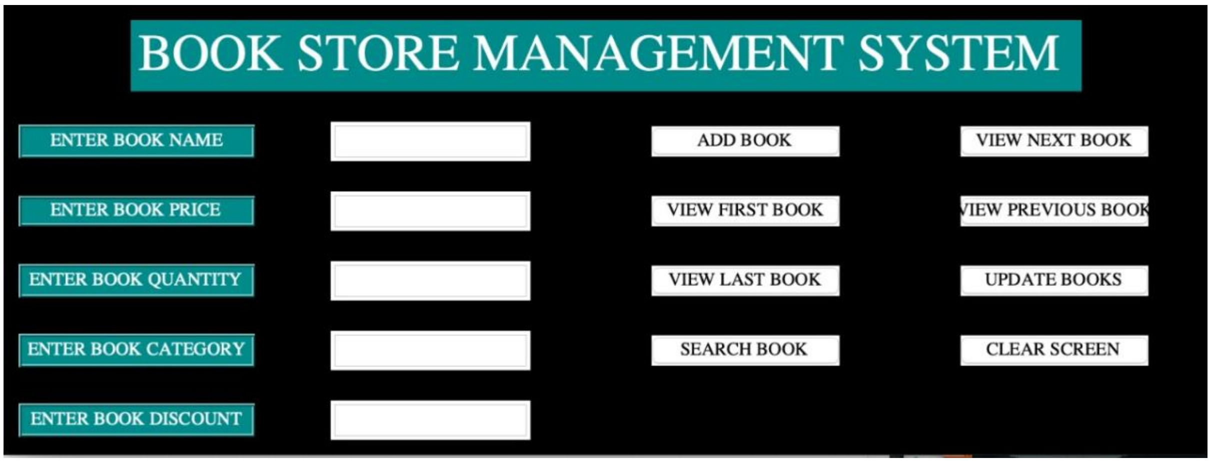
```
button3= Button(root, text="VIEW NEXT BOOK" , bg="white", fg="black", width =20,
font=("Times", 12), command=nextitem)
button4= Button(root, text="VIEW PREVIOUS BOOK", bg="white", fg="black", width =20,
font=("Times", 12),command=previousitem)
button5= Button(root, text="VIEW LAST BOOK", bg="white", fg="black", width =20,
font=("Times", 12),command=lastitem)
button6= Button(root, text="UPDATE BOOKS", bg="white", fg="black", width =20, font=("Times",
12),command=updateitem)
button7= Button(root, text="SEARCH BOOK", bg="white", fg="black", width =20, font=("Times",
12),command=searchitem)
button8= Button(root, text="CLEAR SCREEN", bg="white", fg="black", width=20, font=("Times",
12),command=clearitem)
label0.grid(columnspan=6, padx=10, pady=10)
label1.grid(row=1,column=0, sticky=W, padx=10, pady=10)
label2.grid(row=2,column=0, sticky=W, padx=10, pady=10)
label3.grid(row=3,column=0, sticky=W, padx=10, pady=10)
label4.grid(row=4,column=0, sticky=W, padx=10, pady=10)
label5.grid(row=5,column=0, sticky=W, padx=10, pady=10)
entry1.grid(row=1,column=1, padx=40, pady=10)
entry2.grid(row=2,column=1, padx=10, pady=10)
entry3.grid(row=3,column=1, padx=10, pady=10)
entry4.grid(row=4,column=1, padx=10, pady=10)
entry5.grid(row=5,column=1, padx=10, pady=10)
button1.grid(row=1,column=4, padx=40, pady=10)
button2.grid(row=2,column=4, padx=40, pady=10)
button3.grid(row=1,column=5, padx=40, pady=10)
button4.grid(row=2,column=5, padx=40, pady=10)
button5.grid(row=3,column=4, padx=40, pady=10)
button6.grid(row=3,column=5, padx=40, pady=10)
button7.grid(row=4,column=4, padx=40, pady=10)
button8.grid(row=4,column=5, padx=40, pady=10)
root.mainloop()
```



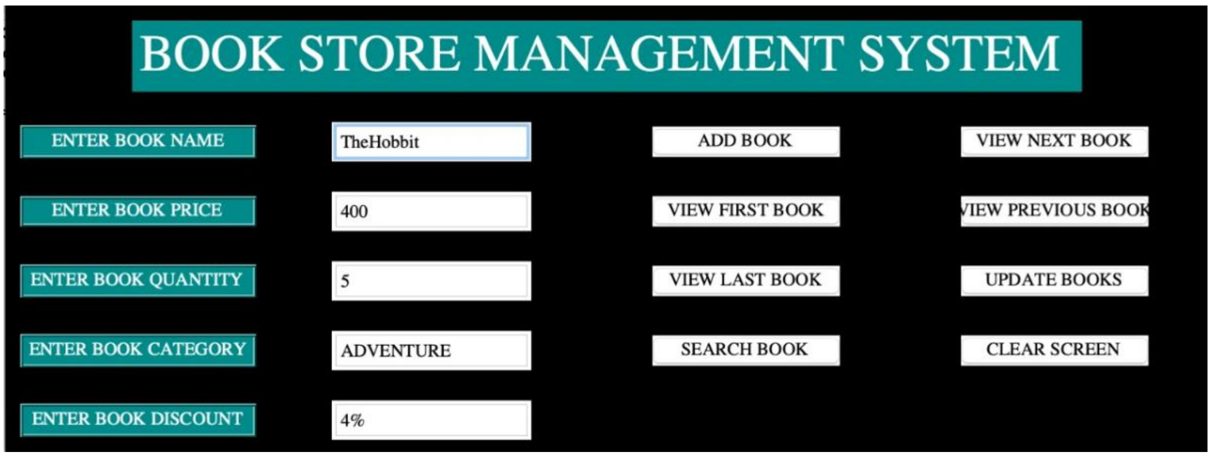
```
TheHobbit 400 5 ADVENTURE 4%
TheThreeMusketeers 450 4 ADVENTURE 7%
LifeofPi 380 7 ADVENTURE 5%
ThePoetsLaureateAnthology 520 3 ANTHOLOGY 10%
ToKillaMockingbird 350 5 Classic 6%
1984 400 5 Classic 7%
RomeoandJuliet 600 6 Classic 4%
Batman:TheDarkKnightReturn 370 7 COMIC 8%
Saga 350 7 COMIC 8%
Superman:TheRebirthDeluxeEdition 350 7 COMIC 8%
Spider-Man:HomecomingMegaMovieStorybook 300 7 COMIC 8%
MarvelZombies Supreme 400 7 COMIC 8%
Batman/Wildcat 370 7 COMIC 8%
KingInBlack:ImmortalHulk 450 10 COMIC 8%
FantasticFour 400 8 COMIC 8%
SherlockHolmes 350 6 Crime 5%
AndThereWereNone 350 6 Crime 5%
MurderonTheOrientExpress 350 6 Crime 5%
Hamlet 450 5 DRAMA 7%
WaitingForGodot 450 5 DRAMA 7%
TheCrucible 450 5 DRAMA 7%
HanselAndGretel 300 8 FAIRYTALE 5%
Rapunzel 300 8 FAIRYTALE 5%
BeautyAndTheBeast 300 8 FAIRYTALE 5%
HarryPotterAndTheSorcerer’sStone 400 10 FANTASY 8%
TheLordOfTheRings 400 10 FANTASY 8%
AGameOfThrones 400 10 FANTASY 8%
TheHungerGames 400 6 SCI-FI 3%
TheMartian 400 6 SCI-FI 3%
Dune 400 6 SCI-FI 3%
```

Records already entered

OUTPUT

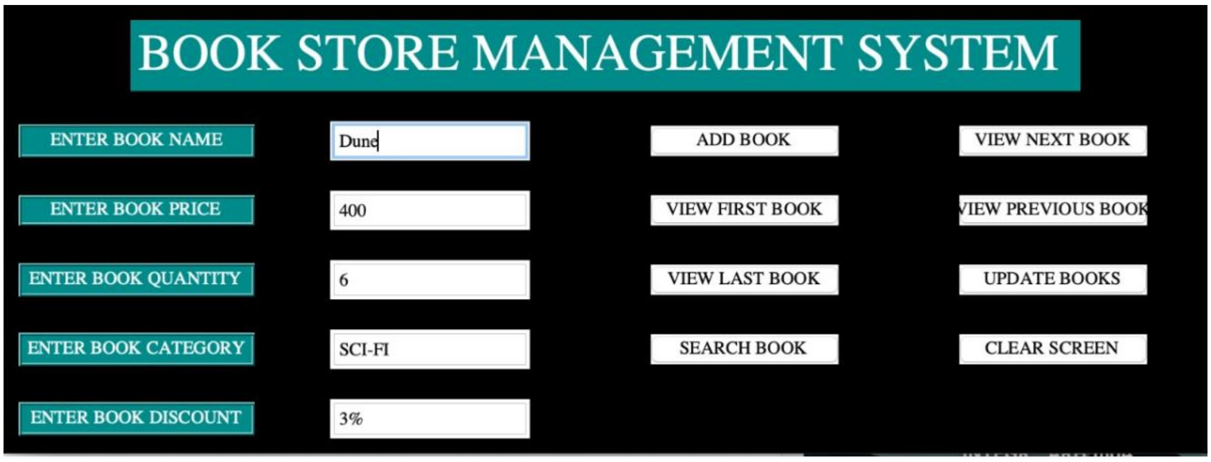


Main view



ENTER BOOK NAME	TheHobbit
ENTER BOOK PRICE	400
ENTER BOOK QUANTITY	5
ENTER BOOK CATEGORY	ADVENTURE
ENTER BOOK DISCOUNT	4%

When you click view first book



ENTER BOOK NAME	Dune
ENTER BOOK PRICE	400
ENTER BOOK QUANTITY	6
ENTER BOOK CATEGORY	SCI-FI
ENTER BOOK DISCOUNT	3%

When you click view last book

TESTING METHODS

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

BLACK BOX TESTING

Black box testing treats the software as a "black box," without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

Specification-based testing aims to test the functionality of software according to the applicable requirements Thus, the tester inputs data into, and only sees the output from, the test object. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behaviour), either "is" or "is not" the same as the expected value specified in the test case. Specification-based testing is necessary, but it is insufficient to guard against certain risks

ADVANTAGES AND DISADVANTAGES

The black box tester has no "bonds" with the code, and a tester's perception is very simple: a code must have bugs. Using the principle, "Ask and you shall receive," black box testers find bugs where programmers don't. But, on the other hand, black box testing has been said to be "like a walk in a dark labyrinth without a flashlight," because the tester doesn't know how the software being tested was actually constructed.

That's why there are situations when (1) a black box tester writes many test cases to check something that can be tested by only one test case, and/or (2) some parts of the back end are not tested at all. Therefore, black box testing has the advantage of "an unaffiliated opinion," on the one hand, and the disadvantage of "blind exploring," on the other.

WHITE BOX TESTING

White box testing, by contrast to black box testing, is when the tester has access to the internal data structures and algorithms (and the code that implement these)

Types of white box testing:-

The following types of white box testing exist:

- API testing - Testing of the application using Public and Private APIs.
- Code coverage - creating tests to satisfy some criteria of code coverage.

For example, the test designer can create tests to cause all statements in the program to be executed at least once.

- Fault injection methods.
- Mutation testing methods.
- Static testing - White box testing includes all static testing.

CODE COMPLETENESS EVALUATION

White box testing methods can also be used to evaluate the completeness of a test suite that was created with black box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested.

Two common forms of code coverage are:

- Function Coverage: Which reports on functions executed and

- Statement Coverage: Which reports on the number of lines executed to complete the test.

They both return coverage metric, measured as a percentage