Visakhapatnam Refinery

# Project Report
# On
# Programming with Python
## LIBRARY MANAGEMENT SYSTEM USING GUI

Vyshnavi Poosarla,

Baba Institute of Technology and Sciences,

Computer Science Department.

# CONTENTS

| SNO | TOPICS | PGNO |
|---|---|---|
| 1 | Project Objective | 01 |
| 2 | Project Tools | 01-05 |
| 3 | Modules | 05-06 |
| 4 | Packages | 06 |
| 5 | Introduction To Project | 07 |
| 6 | Explanation of Project | 08 |
| 7 | Source Code | 09-13 |
| 8 | Output | 14-16 |
| 9 | Conclusion | 17 |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## Project Objective

The main objective of the project is to produce a Library Management System by using a Python Programming Language that utilizes tkinter library to create a Graphical User Interface (GUI). The main goal of the project is to simplify the process of managing books. This source code includes a various types of methods and variables which are used in handling different types of features such as Login, Registration, Adding Books, Issuing Books, and Returning Books. This project is used in real-world also as so many websites are available in the market regarding Libraries so that anybody can browse in the internet and make use of the books in the library.

## Project Tools

- **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

- ### 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.

- ### Why Python?
  - Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
  - Python has a simple syntax similar to the English language.
  - Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
  - 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.

## ➢ Good to Know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is 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.
- 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.

## ➢ COMMENTS:

- Comments can be used to explain Python code
- Comments can be used to make the code more readable.
- Comments can be used to prevent execution when testing code.

➢ Creating a Comment:

- Comments starts with a #,''' COMMENTS ''' and Python will ignore them:
- #This is a comment
- print("Hello, World!")

## ➢ Variables

- Python Variable is containers that store values. Python is not “statically typed”. We do not need to declare variables before using them or declare their type. A variable is created the moment we first assign a value to it. A Python variable is a name given to a memory location. It is the basic unit of storage in a program.
- eg: a = 6
- Here 'a' is the variable.

## ➢ DataTypes

- n programming, data type is an important concept.
- Variables can store data of different types, and different types can do different things.
- Python has the following data types built-in by default, in these categories:

| | |
|---|---|
| ∗ Text Type: | str |
| ∗ Numeric Types: | int, float, complex |
| ∗ Sequence Types: | list, tuple, range |
| ∗ Mapping Type: | dict |

- Set Types: set, frozenset
- Boolean Type: bool
- Binary Types: bytes, bytearray, memoryview
- None Type: NoneType

## Strings

- Strings in python are surrounded by either single quotation marks, or double quotation marks.
- 'hello' is the same as "hello".
- You can display a string literal with the print() function:
- print("Hello")
- print('Hello')

## Lists

- Lists are used to store multiple items in a single variable.
- Create a List:
- books = ["J.P Morgan", "Quantum Physics", "Jungle Book"]
- print(books)
- Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.
- Lists are created using square brackets []

## Python Conditions and If statements

- Python supports the usual logical conditions from mathematics:
- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b
- These conditions can be used in several ways, most commonly in "if statements" and loops.
- An "if statement" is written by using the if keyword.
- The elif keyword is Python's way of saying "if the previous conditions were not true, then try this condition".
- The else keyword catches anything which isn't caught by the preceding conditions.

## Python For Loops

- A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
- This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

- With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

## ➢ Python Functions

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.
- ∗ Creating a Function
- ∗ In Python a function is defined using the def keyword:
- ∗ Calling a Function
- ∗ To call a function, use the function name followed by parenthesis:
- ✓ Arguments
- Information can be passed into functions as arguments.
- Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
- The following example has a function with one argument (fname). When the function is called, we pass along a first name, which is used inside the function to print the full name:

## ➢ Arrays

- Arrays are used to store multiple values in one single variable
- **Note:** We use LISTS as ARRAYS, however, to work with arrays in Python you will have to import a library, like the NumPy library.

## ➢ Python Classes/Objects

- Python is an object oriented programming language.
- Almost everything in Python is an object, with its properties and methods.
- A Class is like an object constructor, or a "blueprint" for creating objects.

- Create a Class
- To create a class, use the keyword class:

- Create Object
- Now we can use the class named MyClass to create objects:

## ➢ The _init_() Function

- To understand the meaning of classes we have to understand the built-in _init_() function.
  All classes have a function called _init_(), which is always executed when the class is being initiated.
- Use the _init_() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

- **Note:** The _init_() function is called automatically every time the class is being used to create a new object.

## ➢ The _str_() Function

- The _str_() function controls what should be returned when the class object is represented as a string.
- If the _str_() function is not set, the string representation of the object is returned.

## ➢ Object Methods:

- Objects can also contain methods. Methods in objects are functions that belong to the object.

## ➢ The self Parameter:

- The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.
- It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class:

- **Note:** The self parameter is a reference to the current instance of the class, and is used to access variables that belong to the class.

## ➢ The pass Statement:

- Class definitions cannot be empty, but if you for some reason have a class definition with no content, put in the pass statement to avoid getting an error.

## ➢ What is PIP?

- PIP is a package manager for Python packages, or modules if you like.

## ➢ What is a Package?

- A package contains all the files you need for a module.
- Modules are Python code libraries you can include in your project.

**Using a Package:**

- Once the package is installed, it is ready to use.
- Import the "camelcase" package into your project.

## ➢ Modules

- In this article, we will cover all about Python modules, such as How to create our own simple module, Import Python modules, From statements in Python, how we can use the alias to rename the module, etc. A Python module is a file containing Python definitions and statements. A module can define functions, classes, and variables.
- It is used to save as '.py' file

- Execute Python Syntax:
- As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:
- >>> print("Hello, World!")
- Hello, World!
- By creating a python file on the server, using the .py file extension, and running it in the Command Line:
- "C:\Users\Your Name>python myfile.py"

## ➢ Packages

- Python modules may contain several classes, functions, variables, etc. whereas Python packages contain several modules. In simpler terms, Package in Python is a folder that contains various modules as files.

## ➢ Creating Package:

- Let’s create a package in Python named mypckg that will contain two modules mod1 and mod2. To create this module follow the below steps:
- Create a folder named mypckg.
- Inside this folder create an empty Python file i.e. _init_.py
- Then create two modules mod1 and mod2 in this folder.
- Mod1.py

## ➢ Understanding _init_.py:

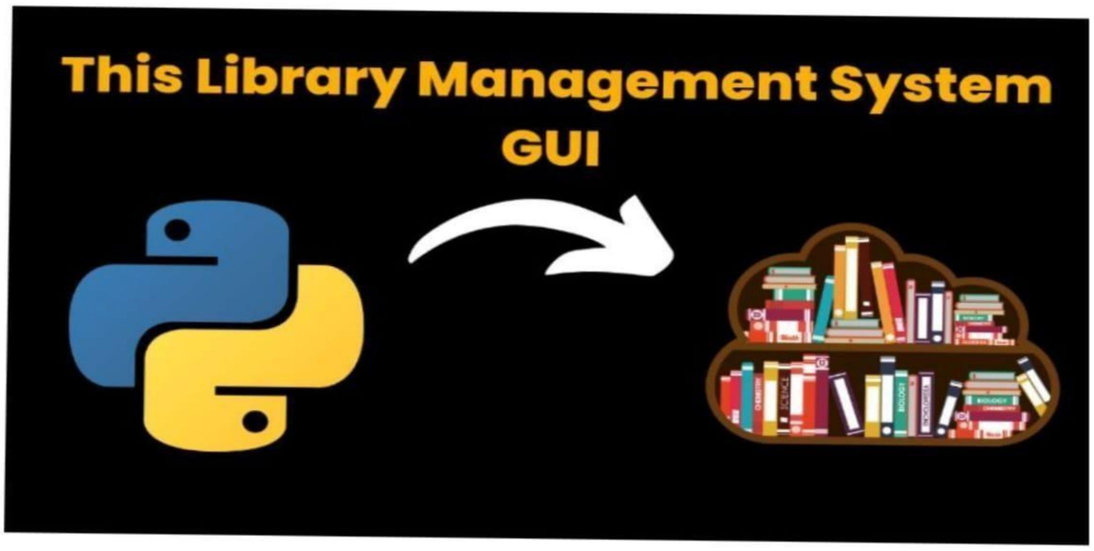
- _init.py helps the Python interpreter recognize the folder as a package. It also specifies the resources to be imported from the modules. If the_init_.py is empty this means that all the functions of the modules will be imported. We can also specify the functions from each module to be made available.
- For example, we can also create the _init_.py file for the above module as:
- _init_.py
- We can import these Python modules using the from…import statement and the dot(.) operator.

## ➢ Creating a GUI program using this Tkinter is simple. For this, programmers need to follow the steps mentioned below:

- Import the module Tkinter
- Build a GUI application (as a window)
- Add those widgets that are discussed above
- Enter the primary, i.e., the main event's loop for taking action when the user triggered the event.

## Introduction To Project

This Library Management System project is a Python-based solution that utilizes the tkinter library to create a graphical user interface (GUI). Its main goal is to simplify and streamline the process of managing books and library members. The code includes a class named “Library Management,” which holds several methods and variables that handle the different features of the system, including login, registration, adding books, removing books, issuing books, and returning books. Although this implementation is basic, it offers room for future enhancements and upgrades.



## Explanation Of  The Project

1. ‘import tkinter as tk’: This line imports the tkinter library and creates an alias tk to reference it easily.
2. ‘from tkinter import messagebox’: This line imports the messagebox module from the tkinter library, which provides dialog boxes to display messages.
3. The class ‘LibraryManagement’ is defined, which has a constructor method  init  ‘.

- The ‘ *init* ‘ method creates a window with a title, dimensions, and a background color. It also initializes lists for books and a lend list.
- Labels and entry boxes for username and password are created, along with Login and Register buttons.
- The *‘login’* method checks the entered username and password against a list of librarians. If the credentials are correct, the login interface is destroyed and the *‘library_management_screen’* method is called.
- The *register* method adds the entered username and password to the list of librarians.
- The *‘library_management_screen’* method creates labels, entry boxes, and buttons for adding, removing, issuing, and viewing books.
- The *‘add_book’* method adds the entered book to the list of books and displays a success message.
- The *‘remove_book’* method removes the entered book from the list of books if it exists and displays a success message, or an error message if it doesn’t exist.
- The *‘issue_book’* method moves the entered book from the list of books to the lend list if it exists and displays a success message, or an error message if it doesn’t exist.
- The *‘view_books’* method displays a message box with a list of all the books in the library.
- The if *name == “ main “:* block creates a *‘Tk’* object, initializes an instance of
- *‘LibraryManagement’*, and starts the main event loop to display the window.

## SOURCE CODE

```
import tkinter as tk
from tkinter import messagebox


class LibraryManagement:
    def _init_(self, master):
        self.master = master
        self.master.title("Library Management System")
        self.master.geometry("400x400")
        self.master.config(bg='#708090')


        self.books = []
        self.lend_list = []


        # Labels
        self.login_label = tk.Label(self.master, text="Library Management System",
font=("Helvetica", 16), bg='#708090', fg='white')
        self.login_label.pack()
        self.username_label = tk.Label(self.master, text="Username", font=("Helvetica", 12),
bg='#708090', fg='white')
        self.username_label.pack()
        self.username_entry = tk.Entry(self.master, font=("Helvetica", 12))
        self.username_entry.pack()
        self.password_label = tk.Label(self.master, text="Password", font=("Helvetica", 12),
bg='#708090', fg='white')
        self.password_label.pack()
        self.password_entry = tk.Entry(self.master, font=("Helvetica", 12), show="*")
        self.password_entry.pack()


        # Login
        self.login_button = tk.Button(self.master, text="Login", command=self.login,
font=("Helvetica", 12))
```

```
        self.login_button.pack()

        # Register
        self.register_button = tk.Button(self.master, text="Register", command=self.register,
font=("Helvetica", 12))
        self.register_button.pack()

        self.username = ""
        self.password = ""
        self.librarians = []

    def login(self):
        self.username  =  self.username_entry.get()
        self.password = self.password_entry.get()
        for librarian in self.librarians:
            if self.username == librarian[0] and self.password == librarian[1]:
                self.username_entry.delete(0, tk.END)
                self.password_entry.delete(0, tk.END)
                self.login_label.destroy()
                self.username_label.destroy()
                self.username_entry.destroy()
                self.password_label.destroy()
                self.password_entry.destroy()
                self.login_button.destroy()
                self.register_button.destroy()
                self.library_management_screen()
                return
        messagebox.showerror("Error",  "Invalid username or password")

    def register(self):
        self.username = self.username_entry.get()
        self.password = self.password_entry.get()
        self.librarians.append([self.username, self.password])
        self.username_entry.delete(0, tk.END)
        self.password_entry.delete(0, tk.END)
    def library_management_screen(self):
        self.add_book_label = tk.Label(self.master, text="Add Book", font=("Helvetica", 16),
bg='#708090', fg='white')
        self.add_book_label.pack()
        self.add_book_entry = tk.Entry(self.master, font=("Helvetica", 12))
        self.add_book_entry.pack()
        self.add_book_button = tk.Button(self.master, text="Add Book",
command=self.add_book, font=("Helvetica", 12))
        self.add_book_button.pack()
        self.remove_book_label = tk.Label(self.master, text="Remove Book",
font=("Helvetica", 16), bg='#708090', fg='white')
        self.remove_book_label.pack()
        self.remove_book_entry = tk.Entry(self.master, font=("Helvetica", 12))
        self.remove_book_entry.pack()
        self.remove_book_button = tk.Button(self.master, text="Remove Book",
command=self.remove_book, font=("Helvetica", 12))
        self.remove_book_button.pack()
        self.issue_book_label = tk.Label(self.master, text="Issue Book", font=("Helvetica", 16),
bg='#708090', fg='white')
        self.issue_book_label.pack()
        self.issue_book_entry = tk.Entry(self.master, font=("Helvetica", 12))
        self.issue_book_entry.pack()
        self.issue_book_button = tk.Button(self.master, text="Issue Book",
command=self.issue_book, font=("Helvetica", 12))
        self.issue_book_button.pack()
        self.view_books_button = tk.Button(self.master, text="View Books",
command=self.view_books, font=("Helvetica", 12))
        self.view_books_button.pack()
    def add_book(self):
        book = self.add_book_entry.get()
        self.books.append(book)
        messagebox.showinfo("Success", "Book added successfully")
        self.add_book_entry.delete(0, tk.END)

    def remove_book(self):
        book = self.remove_book_entry.get()
        if book in self.books:
            self.books.remove(book)
            messagebox.showinfo("Success", "Book removed successfully")
        else:
            messagebox.showerror("Error", "Book not found")
        self.remove_book_entry.delete(0, tk.END)

    def issue_book(self):
        book = self.issue_book_entry.get()
        if book in self.books:
            self.lend_list.append(book)
            self.books.remove(book)
            messagebox.showinfo("Success", "Book issued successfully")
        else:
            messagebox.showerror("Error", "Book not found")
        self.issue_book_entry.delete(0, tk.END)

    def view_books(self):
        message = "\n".join(self.books)
        messagebox.showinfo("Books", message)

if _name_ == "_main_":
```

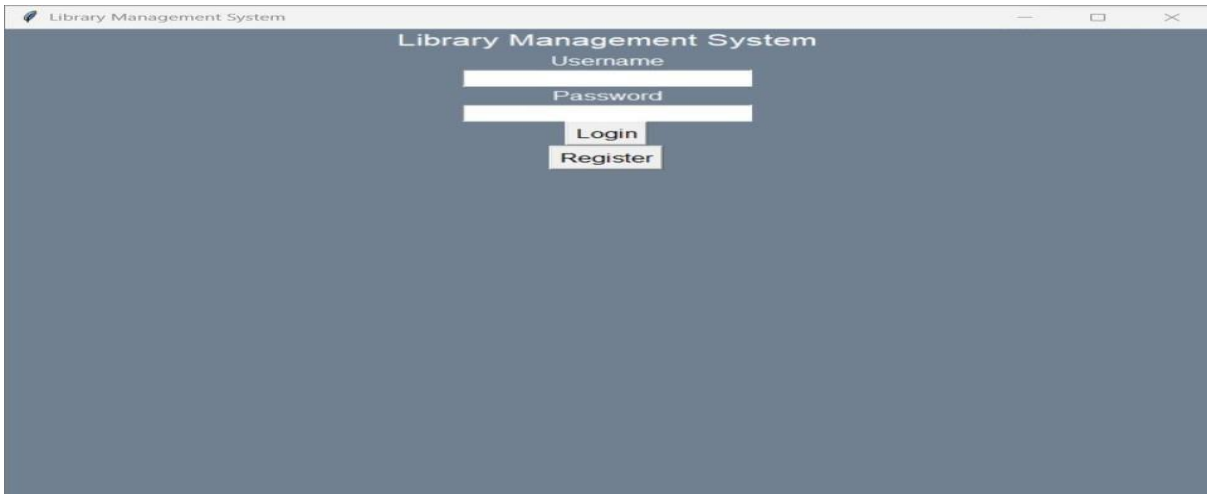
```
root = tk.Tk()

app = LibraryManagement(root)

root.mainloop()
```

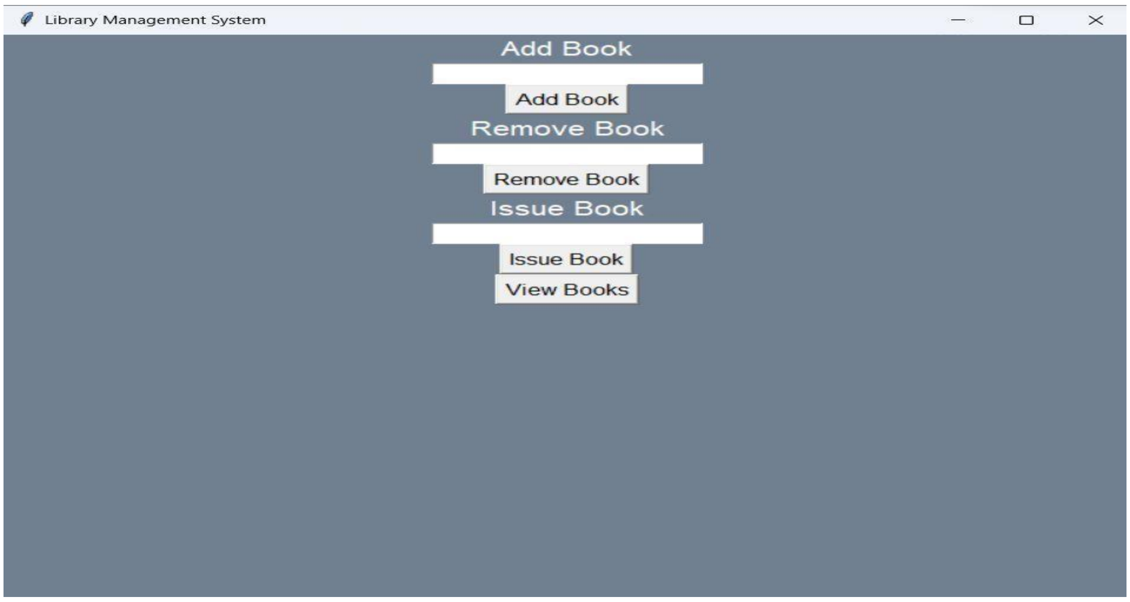
## OUTPUT

In the below interface i.e., Fig 1 , you must enter your Username and Password so that you can Login to your account. If you're a new user you must register in the portal and your credentials will be stored in the Database.

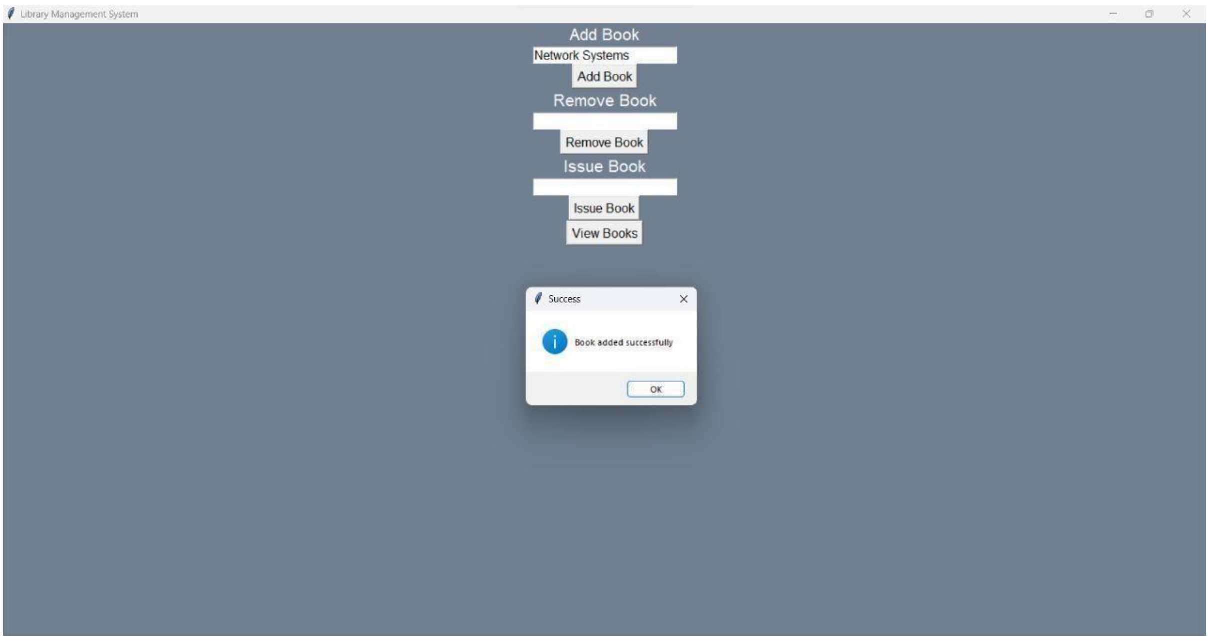


In the below interface i.e., Fig 2 shows that you had Login your account so that it will show you three options namely,

- Add Book
- Remove Book
- Issue Book
- View Books

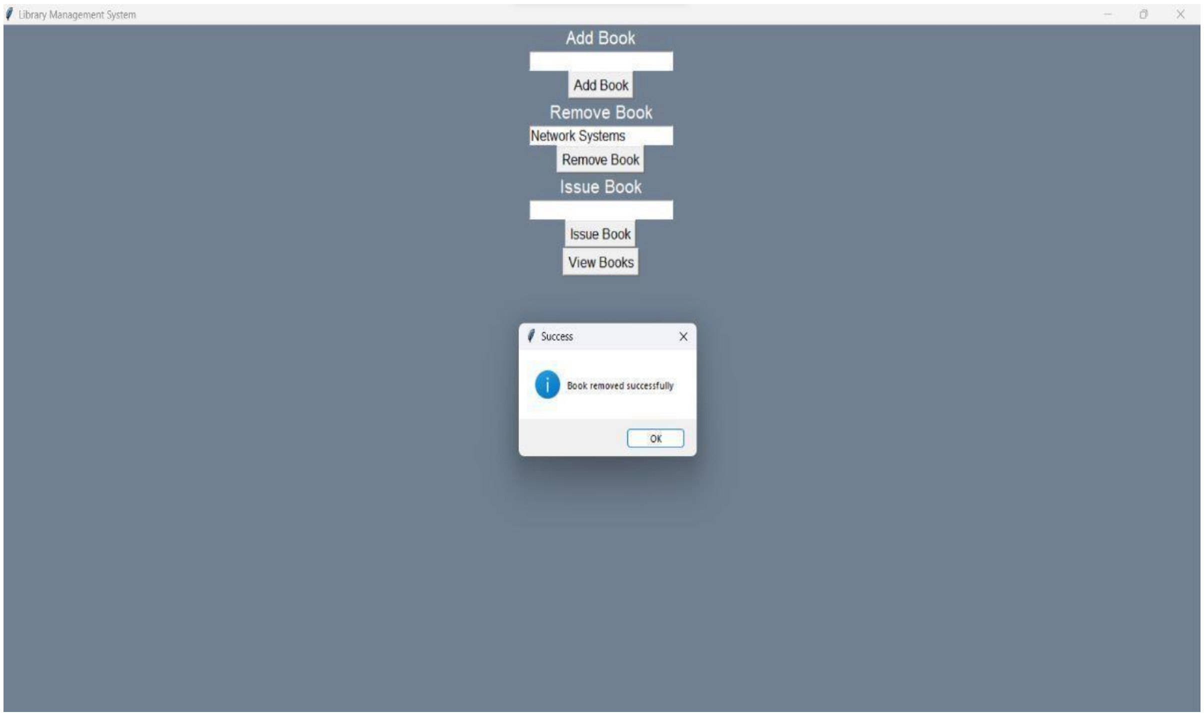


In the below interface, i.e., Fig 3 it is clearly showing that you are adding a book which is NETWORK SYSTEMS which indicates that you are adding that particular book to your portal so that the book is available for you to read whenever you want.



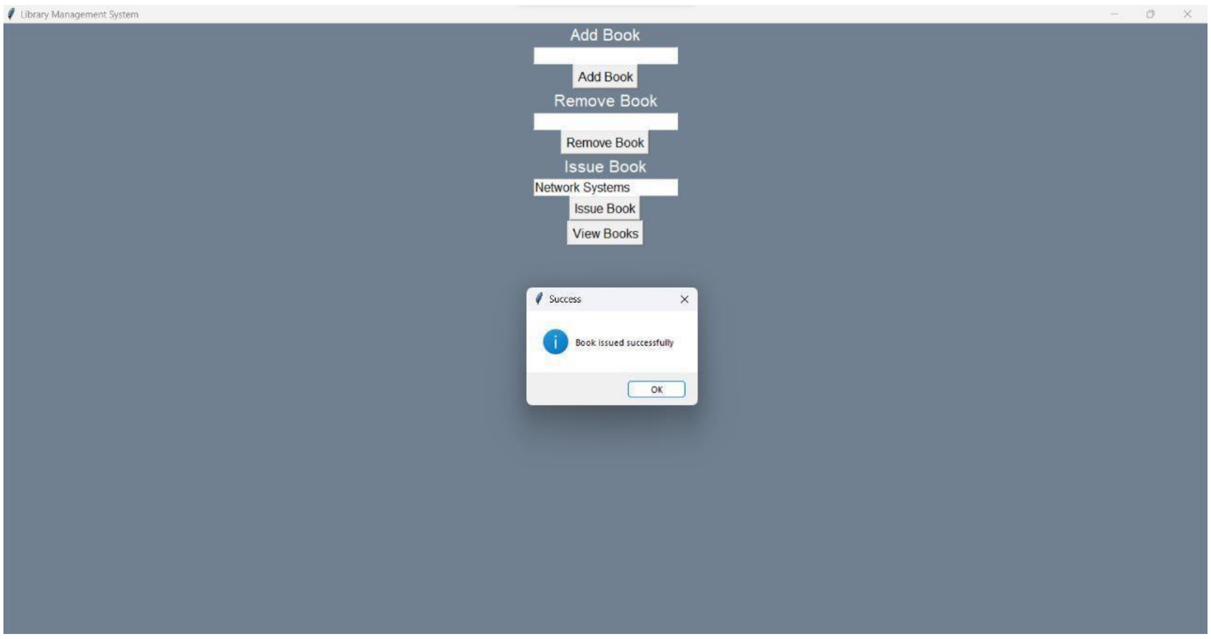
Book added successfully

In the below interface, i.e., Fig 4 it is clearly showing that you are adding a book which is NETWORK SYSTEMS which indicates that you are removing that particular book from your portal.



Book removed successfully

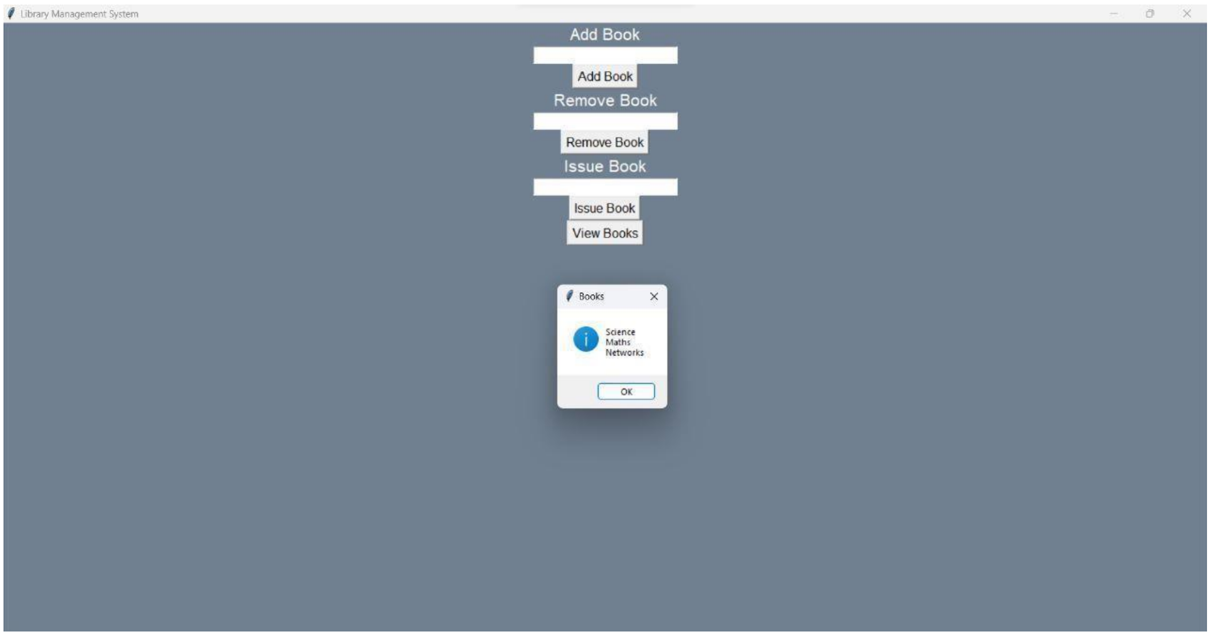
In the below interface, i.e., Fig 5 it is clearly showing that you are issuing a book which is NETWORK SYSTEM which indicates that you are taking that particular book to your portal so that the book is available for you to read whenever you want.



Book issued successfully

In the below interface, i.e., Fig 6 it is clearly showing which are the books available in the portal. For Example, in the Library the books which are available are as follows

- Maths
- Science
- Networks



Science
Maths
Networks

## Conclusion

I had successfully designed a LIBRARY MANAGEMENT SYSTEM USING GUI (Graphical User Interface) using python and tkinter. This project is based upon library books so that anyone can register/Login to their account with the credentials. We can read the books from this website and also delete the books from our respective portal after reading. In this way Library Management System is used.

I would like to thank HPCL Visakha Refinery for giving me this opportunity. In this project I had learned many new things which will be useful in my future.

THANK YOU