READEASE-LIBRARY MANAGEMENT SYSTEM

A MINI-PROJECT REPORT

Submitted by

SHAUN PAUL MOSES	**220701266**
SIHABUTHEEN HAQ P S	**220701278**

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI-602105

2023-2024

BONAFIDE CERTIFICATE

Certified that this project **“READEASE-LIBRARY MANAGEMENT SYSTEM”** is the bonafide work of **“ SHAUN PAUL MOSES (220701266), SIHABUTHEEN HAQ P S (220701278)”** who carried out the project work under mysupervision.

SIGNATURE SIGNATURE
Dr.R.SABITHA Ms.V.JANANEE
Professor and Academic Head, Assistant
Professor(SG) Computer Science and Engineering, Computer
Science and Engineering,Rajalakshmi Engineering College Rajalakshmi
EngineeringCollege, (Autonomous), (Autonomous),
Thandalam,Chennai-602 105 Thandalam,Chennai-602
105

Submitted for the Practical examination to be Held on _

____

_

INTERNAL EXAMINER EXTERNAL EXAMINER

ABSTRACT

Read Ease: A Comprehensive Library Management System

The "Read Ease" project is a sophisticated library management system designed to streamline and enhance the operations of a modern library. This system leverages cutting-edge technology to offer a robust, user-friendly platform that caters to the needs of librarians, staff, and patrons. The primary objective of Read Ease is to facilitate efficient management of library resources, improve user experience, and ensure seamless access to information.

Key features of Read Ease include an intuitive cataloging system, efficient circulation management, and real-time inventory tracking. The cataloging system allows for easy classification and indexing of books and other media, ensuring quick retrieval and accurate record-keeping. Circulation management features support automated check-in/check-out processes, reservation handling, and overdue notifications, significantly reducing manual workload and human error.

Read Ease also incorporates advanced search functionalities, enabling users to effortlessly locate materials through various search parameters such as title, author, genre, and publication date. Additionally, the system supports digital resource integration.

TABLE OF CONTENTS

CHAPTER 1

1.1 . INTRODUCTION

In this program User can perform basic library management operations like issuing books, returning the issued books and displaying records of the is- suedbooks with the user details. Each book in the library has a unique identification number. The user issues the book by entering the book ID and the user details. Each user can issue only one book at a time. When the user returns the issued book, the book is available in the library for issuing again. The record of the issued book with user details can also be viewed.

1.2 OBJECTIVES:

1. **Enhance Operational Efficiency:**

- Develop a system that automates routine tasks such as cataloging, circulation, and inventory management to reduce manual workload and minimize human error.
- Implement a streamlined and user-friendly interface for both librarians and patrons to facilitate quick and easy access to library resources.

2. **Improve User Experience:**

- Integrate advanced search functionalities and personalized recommendations to help users find and access materials effortlessly.
- Create a comprehensive user account management system that allows patrons to track their borrowings, reading history, and receive notifications about due dates and reserved items.

3. **Ensure Data Security and Integrity:**

- Incorporate robust authentication and authorization mechanisms to protect user data and library resources.
- Implement regular data backup procedures and ensure compliance with privacy and data protection standards to safeguard against data loss and unauthorized access.

4. **Support Scalability and Adaptability:**

- Design the system to be scalable to accommodate libraries of various sizes and adaptable to evolving technological advancements and user needs.
- Provide tools for generating detailed reports and analytics to help library administrators make informed decisions and improve overall library management and resource utilization

1.3 MODULES

□ User **Management Module:**

- **User Registration and Authentication:** Manage user sign-ups, logins, password recovery, and authentication processes.
- **User Profiles:** Allow users to update personal information, view borrowing history, and manage account settings.

□ **Catalog Management Module:**

- **Resource Cataloging:** Enable librarians to add, update, and delete library resources such as books, e-books, audiobooks, journals, and other media.
- **Classification and Indexing:** Support various classification systems (e.g., Dewey Decimal, Library of Congress) for efficient organization and retrieval of resources.

□ Search **and Retrieval Module:**

- **Advanced Search:** Provide robust search functionalities allowing users to search by title, author, genre, publication date, ISBN, and other parameters.
- **Filtering and Sorting:** Enable users to filter and sort search results based on different criteria.

CHAPTER -2

SURVEY OF TECHNOLOGIES

2.1 . SOFTWARE DESCRIPTION

The software for the READEASE-Library Management System is developed using Python, which is known for its simplicity and efficiency. The graphical user interface is created using Tkinter and CustomTkinter, libraries that are popular for developing desktop applications in Python. MySQL is employed as the database management system to store and manage data related to book availability. Together, these technologies create a robust and user-friendly application for library Online library .

2.2 LANGUAGES

2.2.1 SQL

Structured Query Language (SQL) is a standard programming language used for managing and manipulating relational databases. In this project, SQL is used extensively to perform various database operations such as creating tables, inserting data, updating records, and fetching data from the database.
The primary SQL operations involved in the project are:

Create Tables: SQL commands are used to define the structure of the database, creating tables to store ticket information.

Insert Data: SQL INSERT statements are used to add initial movie and ticket data into the database.

Update Data: SQL UPDATE statements are used to modify the number of available tickets after a booking is made .

Select Data: SQL SELECT statements are used to retrieve the list of available tickets, which are then displayed in the GUI.

2.2.2 Python

Python is the primary programming language used to develop the Movie Ticket Booking System. It is widely known for its readability and ease of use, making it an ideal choice for both novice and experienced developers. Python's extensive library support allows for rapid development and integration of various functionalities.
Key Python components and libraries used in the project include:

Tkinter and CustomTkinter: These libraries are used to create the graphical user interface (GUI) of the application. Tkinter is the standard GUI toolkit for Python, while CustomTkinter provides additional customization options for creating modern and visually appealing interfaces.
MySQL Connector: This is a Python library that facilitates communication between Python and MySQL databases. It allows the execution of SQL queries from within the Python code, enabling seamless database operations.
File Handling:
Python's built-in file handling capabilities are used to generate and save text files containing booking details, providing users with a tangible record of their transactions.

CHAPTER -3

3.1 REQUIREMENT ANALYSIS

Functional Requirements:

1. **User Registration and Login:**

   ❿ Users should be able to register with personal details (name, email, password).
   ❿ The system must support login/logout functionality.

2. **Book Catalog Management:**

   ❿ Administrators should be able to add, update, and remove book records.
   ❿ Each book record should include title, author, genre, publication year, ISBN, and a summary.

3. **Book Search and Browsing:**

   ❿ Users should be able to search for books by title, author, genre, or ISBN.
   ❿ Search results should include book details and availability status.

4. **Borrowing and Returning Books:**

   ❿ Users should be able to borrow available books, with the system tracking the borrow date and due date.
   ❿ The system should handle the return process and update the book's availability status.

5. **User Account Management:**

   ❿ Users should have a personal dashboard to view their borrowing history and current borrowed books.
   ❿ Notifications for due dates and overdue books should be sent via email.

6. **Book Reservation:**

   ❿ Users should be able to reserve books that are currently checked out.
   ❿ The system should notify users when a reserved book becomes available.

7. **Reviews and Ratings:**

   ❿ Users should be able to leave reviews and ratings for books they have read.
   ❿ Reviews should be visible to other users to help them choose books.

Non-Functional Requirements:

1. **Performance:**

   ❿ The system should handle at least 200 concurrent users with response times for searches and page loads within 3 seconds.

2. **Scalability:**

   ❿ The system should be designed to scale to accommodate a growing number of users and books without major changes.

3. **Reliability:**

   ❿ The system should have an uptime of 99.5%, with reliable data backups to prevent loss of user and book information.

4. **Usability:**

- ❿ The user interface should be simple and intuitive, allowing users to easily find and borrow books.
- ❿ The design should follow accessibility standards to support users with disabilities.

5. **Security:**

- ❿ User data should be protected through secure authentication (e.g., HTTPS, hashed passwords).
- ❿ The system should comply with data privacy regulations (e.g., GDPR, CCPA).

6. **Maintainability:**

- ❿ The system should have a modular architecture to facilitate easy maintenance and updates.
- ❿ Code should be well-documented and follow coding standards for clarity and consistency.

7. **Compatibility:**

- ❿ The system should be compatible with major web browsers (Chrome, Firefox, Safari, Edge).
- ❿ It should have a responsive design to ensure usability on both desktop and mobile devices.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

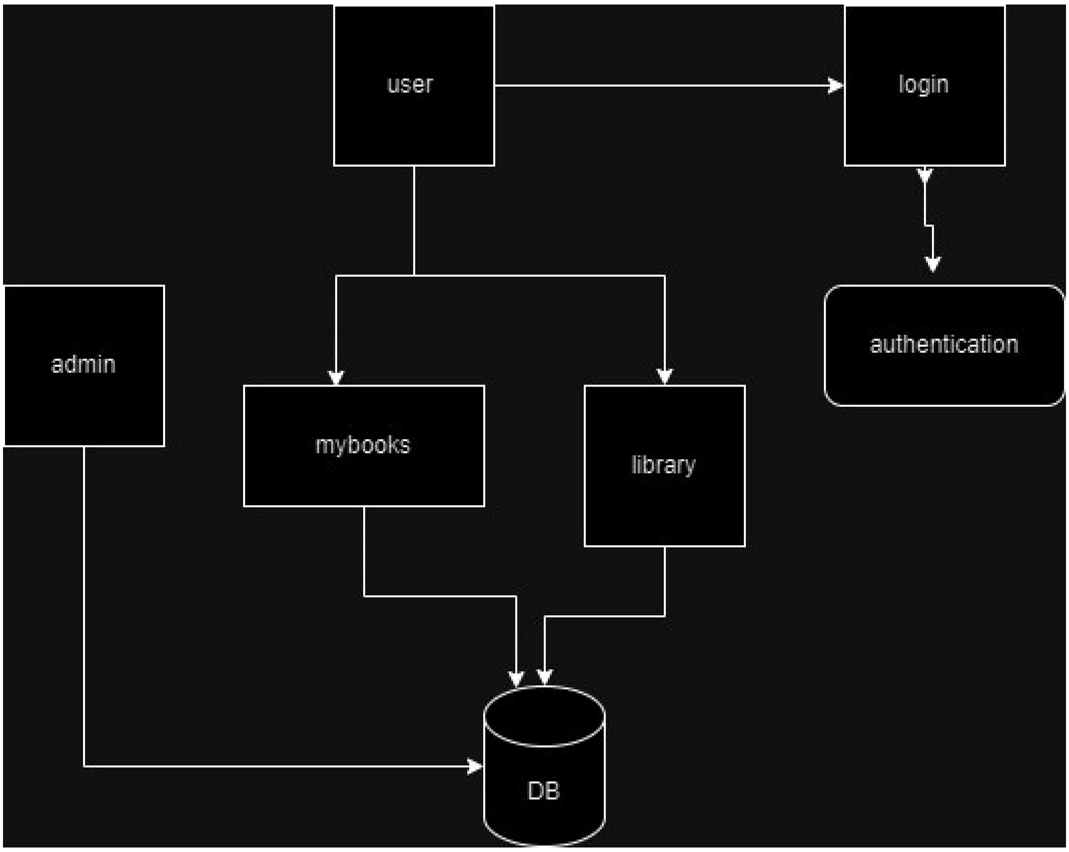
HARDWARE SPECIFICATION

PROCESSOR : INTEL i3
MEMORY SIZE : 4GB
HDD : 256GB

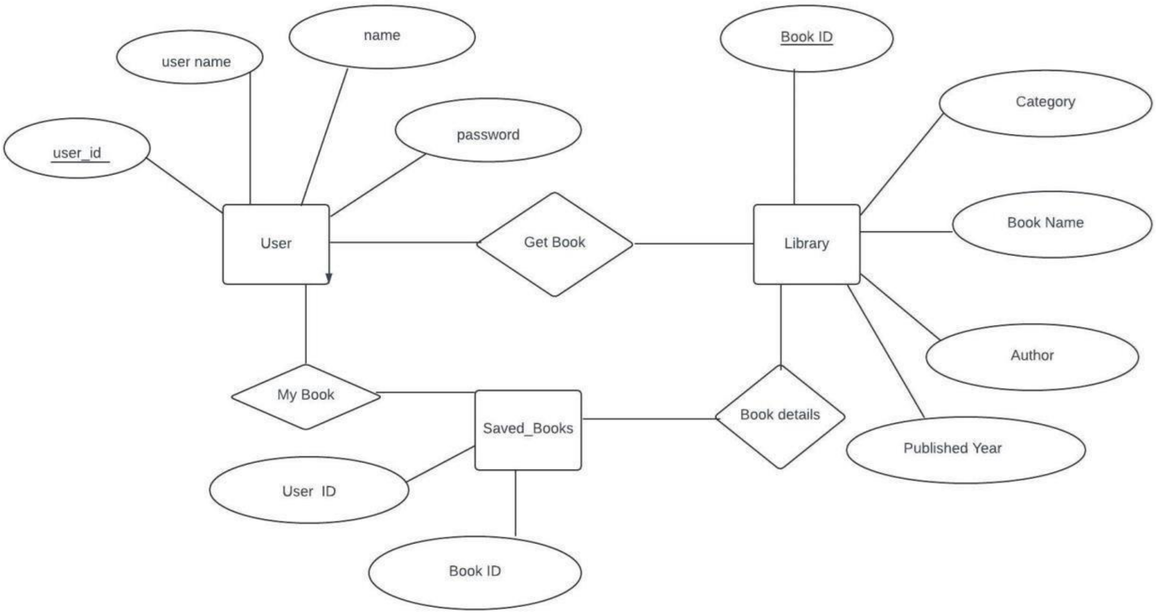
SOFTWARE SPECTFICATION

OPERATING SYSTEM : WINDOWS 11
GUI INTERFACE : PYTHON
BACKEND : MY SQL

3 .3 ARCHITECTURE DIAGRAM



3.4 ER DIAGRAM:



3.5 NORMALIZATION

Normalization Steps:

1. **First Normal Form (1NF):**

   - ➓ Ensure each table column contains atomic values and each column contains values of a single type.

2. **Second Normal Form (2NF):**

   - ➓ Ensure the database is in 1NF.
   - ➓ Remove partial dependencies; all non-key attributes should be fully functionally dependent on the primary key.

3. **Third Normal Form (3NF):**

   - ➓ Ensure the database is in 2NF.
   - ➓ Remove transitive dependencies; non-key attributes should not depend on other non-key attributes.

Normalized Database Schema for Readease:

Table: Users

- ➓ **user_id** (Primary Key)
- ➓ username
- ➓ password
- ➓ email
- ➓ role (e.g., student, faculty, librarian)

Table: Books

- ➓ **book_id** (Primary Key)
- ➓ title
- ➓ author
- ➓ genre
- ➓ publication_year
- ➓ isbn

Table: BorrowedBooks

- ➓ **borrow_id** (Primary Key)
- ➓ book_id (Foreign Key referencing Books.book_id)
- ➓ user_id (Foreign Key referencing Users.user_id)
- ➓ borrow_date
- ➓ due_date
- ➓ return_date

Table: ReservedBooks

- ➓ **reservation_id** (Primary Key)
- ➓ book_id (Foreign Key referencing Books.book_id)
- ➓ user_id (Foreign Key referencing Users.user_id)
- ➓ reservation_date

Table: Reviews

- ➓ **review_id** (Primary Key)

- book_id (Foreign Key referencing Books.book_id)
- user_id (Foreign Key referencing Users.user_id)
- rating
- review_text
- review_date

Table: Categories

- **category_id** (Primary Key)
- category_name

Table: BookCategories

- **book_category_id** (Primary Key)
- book_id (Foreign Key referencing Books.book_id)
- category_id (Foreign Key referencing Categories.category_id)

Explanation of the Normalized Schema:

1. **Users Table:** Stores information about users, ensuring that each user's details are atomic and unique.
2. **Books Table:** Stores details about books, ensuring that each book's attributes are atomic and unique.
3. **BorrowedBooks Table:** Tracks borrowed books, linking books and users with borrow and return details.
4. **ReservedBooks Table:** Manages book reservations, linking books and users with reservation dates.
5. **Reviews Table:** Stores user reviews and ratings for books, ensuring each review is unique and associated with a specific user and book.
6. **Categories Table:** Defines book categories to allow for categorization of books.
7. **BookCategories Table:** Implements a many-to-many relationship between books and categories, allowing books to belong to multiple categories

Benefits of Normalization:

- Reduced data redundancy: Eliminating the repeated book_name minimizes storagespace and reduces the risk of inconsistencies.
- Improved data integrity: Updates to book names only need to be made in the Library table, ensuring consistency across the system.
- Simpler maintenance: The database structure becomes clearer and easier to manage.

While this is a minor improvement, it demonstrates the principles of data normalization. In more complex scenarios, normalization can significantly improve database efficiency and maintainability.

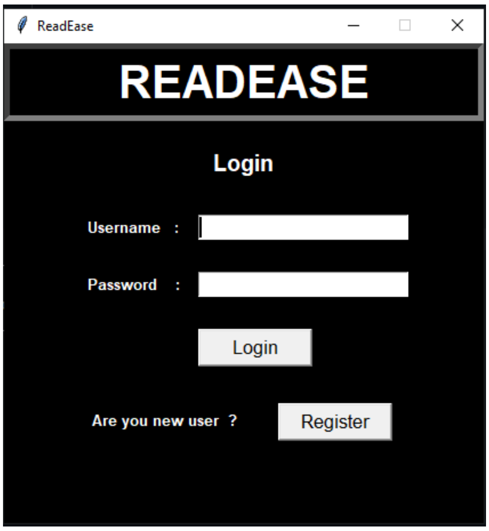
CHAPTER – 4

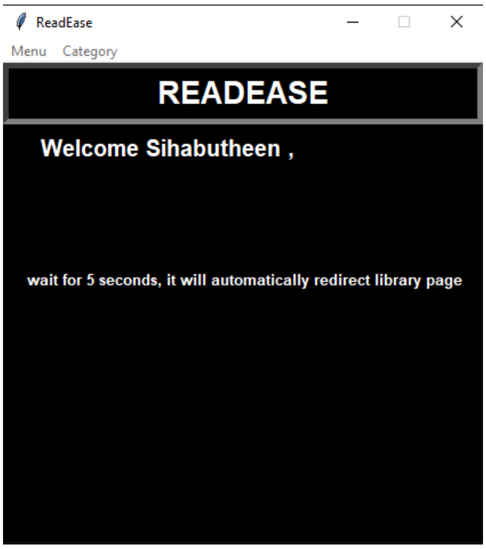
PROGRAM : (Python)

```
from tkinter import *
from tkinter import messagebox
import mysql.connector
from tkinter import ttk
from tkinter import font

class  ReadEase:

    def __init__(self,root):
        window =self.root=root
        window.geometry("420x420")
        window.resizable(False,False)

        window.title("ReadEase")
        self.menu_visible = FALSE;
        self.login_register_frame  =  Frame(self.root)
        self.login_register_frame.config(bg='black')
        self.login_register_frame.pack(fill=BOTH  ,  expand=TRUE)

        self.main_widget_frame = Frame(self.root)
        self.main_widget_frame.config(bg ='black')
        self.main_widget_frame.pack_forget()
        self.id = None

        self.create_login_register()

        self.conn  =  mysql.connector.connect(
            host='localhost',
            user='root',
            password='Sihab@8117',
            database='readease'
        )
        self.c  =  self.conn.cursor()
        self.c.execute('create table if not exists user (id int AUTO_INCREMENT primary key,name
varchar(16),username varchar(16) ,password varchar(8) not null)')
        self.c.execute('create table if not exists saved_books(id int ,product_id int ,book_name
varchar(50) ,foreign key (id) references user(id))')
```

```
        self.c.execute('create table if not exists library(product_id int AUTO_INCREMENT
primary key ,book_name varchar(50) ,author varchar(50) ,category varchar(50) ,year int)')

    def create_login_register(self):

        self.create_readease_panel_login()
        self.login_page = Label(self.login_register_frame ,text =('Login'),
                              font=('Comic Sans',15,'bold'),
                              fg='white',
                              bg='black')
        self.username = Label(self.login_register_frame ,text =('Username :'),
                              font=('Arial',10,'bold'),
                              fg='white',
                              bg='black')
        self.username_entry =Entry (self.login_register_frame , font =("Arial"))
        self.password_label = Label (self.login_register_frame ,text='Password                    :',
                                  font=('Arial',10,'bold'),
                                  bg='black',
                                  fg='white')
        self.password_entry =Entry (self.login_register_frame , font =("Arial") ,show="*")
        self.login = Button(self.login_register_frame ,
                    text ='Login',
                    font=("Comic Sans" ,13),
                    width= 10,
                    command = self.login_to
                    )
        self.register =Button(self.login_register_frame ,
                    text ='Register',
                    font=("Comic Sans" ,13),
                    command=self.show_register,
                    width=10)
        self.new_register = Label (self.login_register_frame ,text=' Are you new user ? ',
                                  font=('Arial',10,'bold'),
                                  bg='black',
                                  fg='white')
        self.readease_label.pack()
        self.login_page.place(x=180,y=90)
        self.username.place(x=70 , y=150)
        self.username_entry.place(x=170,y=150)
        self.password_label.place(x=70 ,y=200)
        self.password_entry.place(x=170,y=200)
        self.login.place(x= 170 ,y= 250 )
        self.new_register.place(x=70 ,y=319)
        self.register.place(x=240 ,y=315)
        self.register_page =Label(self.login_register_frame ,
                           text='Register',
                           font=('Arial',14,'bold'),
                           fg='white',
                           bg='black')
        self.name_label =Label(self.login_register_frame ,
                           text='Name                                        :', font=('Arial',10,'bold'),
                           fg='white',
                           bg='black')
        self.name_entry =Entry (self.login_register_frame , font =("Arial"))
        self.repassword_label =Label(self.login_register_frame ,
                           text='Re Enter Password :',
                              font=('Arial',10,'bold'),
                              fg='white',
                              bg='black',
                              )
            self.repassword_entry =Entry (self.login_register_frame , font =("Arial"),show="*")
            self.register_register =Button(self.login_register_frame ,
                            text ='Register',
                            font=("Comic Sans" ,13),
                            command =self.Register,
                            width=12
                            )
            self.back_to_login =Button(self.login_register_frame ,
                            text ='Back to Login',
                            font=("Comic Sans" ,13),
                            command=self.show_login,
                            width=12

                            )
            self.back_to_login_label =Label(self.login_register_frame  ,
                              text='Back to Login page ->',
                              font=('Arial',10,'bold'),
                              fg='white',
                              bg='black',
                              )
    def    create_readease_panel_login(self):
        self.readease_label = Label(self.login_register_frame ,
                  text="READEASE" ,
                    font =('Arial' ,30,'bold') ,
                     fg='white' ,bg='black' ,
                     relief=SUNKEN ,
                     bd=10,
                     padx=1000)
    def login_to(self):
        self.welcome_check=True
        user =self.username_entry.get()
        password = self.password_entry.get()
        self.c.execute('SELECT username, password FROM user WHERE username = %s',
(user,))
        userpass = self.c.fetchone()
        self.c.execute('Select id from user WHERE username = %s', (user,))
        id = self.c.fetchone()
        if userpass:
            u , p = userpass
            if u==user and p == password:
                self.id = id
                self.create_main_widget()
            else :
                messagebox.showerror("Error", "Invalid email or password please try again")
                self.des_detail()
        else :
            messagebox.showerror("Error", "please Register !!!")
            self.des_detail()

    def Register(self) :
        name = self.name_entry.get()
        uname = self.username_entry.get()
        password = self.password_entry.get()
        repassword  =self.repassword_entry.get()

        self.c.execute('select username from user where username = %s',(uname,))
        result=self.c.fetchone()

        if password != repassword :
            messagebox.showerror("Password not matched", "Password Must be matched ! please
try again .")
        elif result is not None :
            messagebox.showerror("Try another username ","Username already Exists Try another
username !")
            self.des_detail()
        elif name ==" or uname ==" or password ==" or repassword==":
            messagebox.showerror("Empty field found !" ,"All the fields are reqired .please fill
!!!")
        else:
            try:
                self.c.execute('insert into user(name,username ,password) values (%s,%s,%s)'
,(name,uname,password,))
                self.conn.commit()
                messagebox.showinfo("Success", "Registration successful! Please login.")
            except mysql.connector.Error as err:
                messagebox.showerror("Error", str(err))
                self.des_detail()


    def   show_login(self):
        self.login_register_frame.pack(fill=BOTH , expand=TRUE)
        self.main_widget_frame.pack_forget()

        for widget in self.login_register_frame.winfo_children():
           widget.pack_forget()
           widget.place_forget()
        self.root.config(menu=FALSE)
        self.create_readease_panel_login()

        self.readease_label.pack()
        self.login_page.place(x=180,y=90)
        self.username.place(x=70 , y=150)
        self.username_entry.place(x=170,y=150)
        self.password_label.place(x=70 ,y=200)
        self.password_entry.place(x=170,y=200)
        self.login.place(x= 170 ,y= 250 )
        self.new_register.place(x=70 ,y=319)
        self.register.place(x=240 ,y=315)



    def show_register(self):
        for widget in self.login_register_frame.winfo_children():
           widget.pack_forget()
           widget.place_forget()
        self.readease_label.pack()

        self.register_page.place(x=170,y=80)

        self.name_label.place(x=60 ,y=120)
        self.name_entry.place(x=  200,y=120)
        self.username.place(x=60 ,y=160)
        self.username_entry.place(x= 200,y=160)
        self.password_label.place(x=60 ,y=200)
        self.password_entry.place(x= 200,y=200)
        self.repassword_label.place(x=60 ,y=240)
        self.repassword_entry.place(x= 200,y=240)
        self.register_register.place(x=250 ,y= 290 )
        self.back_to_login_label.place(x=80 ,y=355)
        self.back_to_login.place(x=250 ,y=350)

    def des_detail(self):
        self.username_entry.delete(0,'end')
        self.password_entry.delete(0,'end')
        self.repassword_entry.delete(0,'end')
        self.name_entry.delete(0,'end')



    def show_message(self):
        try:
            self.c.execute('select name from user where id =%s',self.id)
            name =self.c.fetchone()
            name = name[0]
        except mysql.connector.Error as err:
            messagebox.showerror("Error", str(err))
        if self.welcome_check:

            self.nav=Label(self.main_widget_frame ,
                           text=f" wait for 5 seconds, it will automatically redirect library page",
                           font=('Arial',10,'bold'),
                           fg='white',
                           bg='black',
                           pady=100
                           )
            self.welcome =Label(self.main_widget_frame ,
                           text=f"Welcome {name} ,",
                           font=('Arial',15,'bold'),
                           fg='white',
                           bg='black'
                           )

            self.nav.place(x=10,y=80)
            self.welcome.place(x=30 ,y=60)
            self.root.after(5000,self.show_library)

    def create_main_widget(self) :

        self.des_detail()
        self.login_register_frame.pack_forget()
        self.login_register_frame.place_forget()
        self.main_widget_frame.pack(fill =BOTH ,expand=TRUE)
        self.main_widget_frame.config(bg='black')

        self.readease_label = Label(self.main_widget_frame ,
                 text="READEASE" ,
                  font =('Arial' ,20,'bold') ,
                   fg='white' ,bg='black' ,
                   relief=SUNKEN ,
                    bd=10,
                    padx = 125,
                    )
        self.readease_label.place(x=0,y=0)

        try:
            self.c.execute('select name from user where id =%s',self.id)
            name =self.c.fetchone()
            name = name[0]
        except mysql.connector.Error as err:
            messagebox.showerror("Error", str(err))

        self.menu_bar = Menu(self.root)

        self.root.config(menu=self.menu_bar)
        global option_state
        option_state = False


        self.toggle_menu = Menu(self.menu_bar, tearoff=0 ,bg='black' ,fg='white')
        self.menu_bar.add_cascade(label="Menu", menu=self.toggle_menu )
        self.toggle_menu.add_command(label="profile" ,background='black' ,foreground='white'
,command =self.show_profile )
        self.toggle_menu.add_command(label="library",background='black' ,foreground='white'
,command=self.show_library )
        self.toggle_menu.add_command(label="saved  books",background='black'
,foreground='white' ,command=self.show_saved_books )
        self.toggle_menu.add_command(label="change password",background='black'
,foreground='white',command=self.show_change_password)
        self.toggle_menu.add_command(label="logout",background='black' ,foreground='white'
,command=self.show_login)

        self.create_profile()
        self.create_library()
        self.create_saved_books()
        self.create_change_password()

        self.category = Menu(self.menu_bar, tearoff=0 ,bg='black' ,fg='white')
        self.menu_bar.add_cascade(label="Category", menu=self.category )

        if self.welcome_check:
            self.menu_bar.entryconfig(2, state="disabled")
            self.menu_bar.entryconfig(1, state= "disabled")
            self.show_message()

    def hide_main_frame(self):
        self.welcome_check = False
        for widget in self.main_widget_frame.winfo_children():
          widget.pack_forget()
          widget.place_forget()


    def cat_click(self,cat ,list):
        self.search_library_entry.delete(0,'end')
        if list == '0':
            self.insert_listbox(cat)
        elif list =='1':
            self.insert_saved_listbox(cat)
        self.deselect_all()

    def insert_to_category(self):
        self.category.add_command(label='all ' ,background='black' ,foreground='white'
,command=lambda:self.cat_click("all",'0') )
        self.c.execute('select DISTINCT category from library')
        while True:
            row = self.c.fetchone()
            if row is None:
                break
            self.category.add_command(label=row ,background='black' ,foreground='white'
,command=lambda cat = row[0]: self.cat_click(cat,'0') )

    def insert_to_saved(self):

        self.category.add_command(label='all ' ,background='black' ,foreground='white',
command=lambda: self.cat_click("all",'1'))
        self.c.execute('select DISTINCT product_id from saved_books where id = %s',self.id)
        rows = self.c.fetchall()



        rows = tuple(item[0] for item in rows)
        placeholders = ', '.join(['%s'] * len(rows))
        query = f'SELECT category FROM library WHERE product_id IN ({placeholders})'
        self.c.execute(query, rows)
        results = self.c.fetchall()
        results = tuple(item[0] for item in results)
        results= set(results)
        results=tuple(results)

        for row in results:
            self.category.add_command(label=row ,background='black' ,foreground='white' ,
command=lambda row = row : self.cat_click(row,'1'))




    def change_password(self):
        id=self.id[0]
        old_password = self.old_password_entry.get()
        new_password = self.new_password_entry.get()
        try :
            self.c.execute('select password from user where id=%s',(id,))
        except mysql.connector.Error as err:
            messagebox.showerror("Error", str(err))

        check_old_password = self.c.fetchone()



        if new_password == '' or old_password=='':
            messagebox.showerror("Password field Empty"," password field is empty please
provide password")
        elif check_old_password[0] != old_password :
            messagebox.showerror("incorrect password" ,"Old password not matched !")
        else:
            try:
                self.c.execute('update user set password =%s where id =%s',(new_password,id))
                self.conn.commit()
                self.show_profile()
                messagebox.showinfo("Password changed","Password changed successfully")

            except mysql.connector.Error as err:
                messagebox.showerror("Error", str(err))
        self.old_password_entry.delete(0,'end')
        self.new_password_entry.delete(0,'end')



    def deselect_all(self):
        self.book_listbox.selection_clear(0,END)
        self.listbox_saved.selection_clear(0,END)


    def delete_listbox(self):
        if self.book_listbox.size()==0:
            messagebox.showinfo("Empty !","Library is Empty please add Books")
            return
        index=self.book_listbox.curselection()
        index=self.book_listbox.get(index)
        if index:
            product_key = int(index.split(":")[0])
            self.c.execute('delete from library where product_id = %s',(product_key,))
            self.c.execute('delete from saved_books where product_id = %s',(product_key,))
            self.conn.commit()
        if self.book_listbox.size()==1:
            messagebox.showinfo("Empty !","Library is Empty please add Books")
        self.deselect_all()
        self.show_library()


    def delete_saved_listbox(self):
        if self.listbox_saved.size()==0:
            messagebox.showinfo("Empty !"," Empty please add Books")
            return
        index=self.listbox_saved.curselection()
        index=self.listbox_saved.get(index)
        if index:
            product_key = int(index.split(":")[0])
            self.c.execute('delete from saved_books where product_id = %s and id
=%s',(product_key,self.id[0],))
            self.conn.commit()
        if self.listbox_saved.size()==1:
            messagebox.showinfo("Empty !","Empty please add Books")
        self.deselect_all()
        self.show_saved_books()


    def show_book_info(self ,product_id):
        self.book_info =Toplevel(self.root)
        self.book_info.geometry("450x350")
        self.book_info.title("book_info new book")
        self.book_info.resizable(False,False)
        self.book_info_window_frame = Frame(self.book_info)
        self.book_info_window_frame.config(bg='black')
        self.book_info_window_frame.pack(fill=BOTH , expand=TRUE)

        self.c.execute('select book_name,author,category,year from library where product_id
    =%s',(product_id,))
        bookname ,author,category,year = self.c.fetchone()

        self.i_book_id =Label(self.book_info_window_frame,
                    text='Book id                         :', font=('Arial',10,'bold'),
                    fg='white',
                    bg='black'
                    )
        self.i_book_name =Label(self.book_info_window_frame,
                    text='Book name                       :', font=('Arial',10,'bold'),
                    fg='white',
                    bg='black'
                    )
        self.i_category =Label(self.book_info_window_frame,
                    text='Book Category                   :', font=('Arial',10,'bold'),
                    fg='white',
                    bg='black'
                    )
        self.i_author =Label(self.book_info_window_frame,
                    text='Author name                     :', font=('Arial',10,'bold'),
                    fg='white',
                    bg='black'
                    )
        self.i_yop =Label(self.book_info_window_frame,
                    text='year of publishing :',
                    font=('Arial',10,'bold'),
                    fg='white',
                    bg='black'
                    )

        self.i_book_id_value =Label(self.book_info_window_frame,
                    text=product_id,
                    font=('Arial',10,'bold'),
                    fg='white',
                    bg='black'
                    )
        self.i_book_name_value =Label(self.book_info_window_frame,
                    text =bookname,
                    font=('Arial',10,'bold'),
                    fg='white',
                    bg='black'
                    )
        self.i_category_value =Label(self.book_info_window_frame,
                    text =category,
                    font=('Arial',10,'bold'),
                    fg='white',
                    bg='black'
                    )
        self.i_author_value =Label(self.book_info_window_frame,
                    text=author,
                      font=('Arial',10,'bold'),
                      fg='white',
                      bg='black'
                      )
        self.i_yop_value=Label(self.book_info_window_frame,
                      text =year,
                      font=('Arial',10,'bold'),
                      fg='white',
                      bg='black'
                      )

        self.i_book_id.place(x=50,y=50)
        self.i_book_name.place(x=50,y=100)
        self.i_category.place(x=50,y=150)
        self.i_author.place(x=50,y=200)
        self.i_yop.place(x=50,y=250)

        self.i_book_id_value.place(x=200,y=50)

        self.i_book_name_value.place(x=200,y=100)
        self.i_category_value.place(x=200,y=150)
        self.i_author_value.place(x=200,y=200)
        self.i_yop_value.place(x=200,y=250)


    def  on_double_click(self,event):
        index  =  self.book_listbox.curselection()
        if index:
            index=self.book_listbox.get(index)
            product_id  =  int(index.split(":")[0])
            self.show_book_info(product_id)
            self.deselect_all()
        else :
            index  =  self.listbox_saved.curselection()
            if index:
                 index=self.listbox_saved.get(index)
                 product_id = int(index.split(":")[0])
                 self.show_book_info(product_id)
                 self.deselect_all()



    def save_user_book(self):
        id=self.id[0]
        index=self.book_listbox.curselection()
        index=self.book_listbox.get(index)
        if index:
            product_key  =  int(index.split(":")[0])
            self.c.execute('select product_id from saved_books where product_id=%s and id
=%s',(product_key,id,))
            check = self.c.fetchone()
            if check == None :
                try:
                    book = index.split(":")[1].strip()
                    self.c.execute('insert  into  saved_books(id  ,product_id,book_name)
values(%s,%s,%s)',(id,product_key,book,))
                    self.conn.commit()
                    messagebox.showinfo("Book saved !","selected book was saved successully !")
                    self.deselect_all()
                except mysql.connector.Error as err:
                    messagebox.showerror("Error", str(err))
            else:
                messagebox.showerror("Already saved !","Selected book was already saved !")
                self.deselect_all()
        self.deselect_all()

    def library_search_command(self):
        entry = self.search_library_entry.get().lower()
        entry = f"{entry.lower()}%"
        try :
            self.c.execute('select distinct(product_id) from library where lower(book_name) LIKE
%s',(entry,))
            books = self.c.fetchall()
            self.insert_listbox(p_id = books)
        except mysql.connector.Error as err:
            messagebox.showerror("Error","No book found !!!")
            self.show_library()
            return


    def saved_search_command(self):
        entry = self.search_saved_entry.get().lower()
        entry = f"{entry.lower()}%"
        try :
            self.c.execute('select distinct(product_id) from saved_books where lower(book_name)
LIKE %s',(entry,))
            books = self.c.fetchall()
            self.insert_saved_listbox(p_id = books)
        except mysql.connector.Error as err:
            messagebox.showerror("Error","No book found !!!")
            self.show_saved_books()
            return

    def create_library(self):
        self.library_page =Label(self.main_widget_frame,
                        text='Library',
                        font=('Comic Sans',13,'bold'),
                        fg='white',
                        bg='black')
        self.save_btn =Button(self.main_widget_frame ,
                          text ='Save',
                          font=("Comic Sans" ,13),
                          width=10,
                          command=self.save_user_book
                          )
        self.add_btn =Button(self.main_widget_frame ,
                          text ='Add',
                          font=("Comic Sans" ,13),
                          width=10,
                          command = self.add_window
                          )
        self.delete_btn =Button(self.main_widget_frame ,
                          text ='Delete',
                          font=("Comic Sans" ,13),
                          width=10,
                            command=self.delete_listbox
                            )
        self.book_listbox = Listbox(self.main_widget_frame,
                                  font=('Comic Sans', 12),
                                  width=45,
                                    height=15,
                                    bg ='#040404',
                                    fg='white')


        self.list_scrollbar = Scrollbar(self.main_widget_frame, orient=VERTICAL,
command=self.book_listbox.yview,bg='gray')
        self.book_listbox.config(yscrollcommand=self.list_scrollbar.set ,
selectmode=EXTENDED)
        self.book_listbox.bind('<Double-Button-1>',  self.on_double_click)

        # create search library
        self.search_library_entry = Entry( self.main_widget_frame ,font = "Arial")
        self.search_library_button = Button(self.main_widget_frame ,
                            text ='Search',
                            font=("Comic Sans" ,11),
                            width=10,
                            command=self.library_search_command
                            )

    def del_item_insert_listbox(self):
        self.book_listbox.delete(0,  'end')


    def del_item_insert_saved_listbox(self):
        self.listbox_saved.delete(0, 'end')


    def insert_listbox(self ,cat ='all' ,p_id = None):

        if p_id != None :
             values = [item[0] for item in p_id]
             placeholders = ', '.join(['%s'] * len(values))
             query = f'SELECT * FROM library WHERE product_id IN ({placeholders})'

        self.library_select_category = cat
        self.create_side_bar_library()

        self.del_item_insert_listbox()
        if cat != 'all':
                if p_id != None :
                     self.c.execute(query,  values)
                else :
                  self.c.execute('Select * from library where category =%s ',(cat,))

        else:
           if p_id != None :
                self.c.execute(query,  values)
           else :
             self.c.execute('Select * from library')

        while True:
            row = self.c.fetchone()
            if row is None:
                break
            index,book_name,author,category,year =row
            self.book_listbox.insert(END,f"{index}:  {book_name}")



    def insert_saved_listbox(self ,cat ='all' ,p_id = None):
        self.saved_select_category = cat
        self.del_item_insert_saved_listbox()
        self.create_side_bar_saved()

        if p_id != None :
             values = [item[0] for item in p_id]
             placeholders = ', '.join(['%s'] * len(values))
             query = f'SELECT product_id,book_name FROM saved_books WHERE product_id
 IN ({placeholders}) AND id =%s'
             values = values + [self.id[0]]
             try :
                 self.c.execute(query , values)
             except mysql.connector.Error as err:
                messagebox.showerror("Error", str(err))
                self.show_saved_books()
        else :
           self.c.execute('Select product_id,book_name from saved_books  where id = %s',self.id)
        rows = self.c.fetchall()
        for row in rows:
            product_key,bookname  = row
            if cat != 'all':
                self.c.execute('select category from library where product_id =%s',(product_key,))
                result = self.c.fetchone()
                if  cat  ==  result[0]:
                    self.listbox_saved.insert(END,f"{product_key}:  {bookname}")

            else :
               self.listbox_saved.insert(END,f"{product_key}:   {bookname}")


    def add_to_library(self):
        self.deselect_all()
        book_name = self.book_name_entry.get()
        category = self.category_entry.get()
        author= self.author_name_entry.get()
        yop  = self.year_selector.get()


        if book_name =='':
            messagebox.showerror("book name field is empty please fill now !")
        elif category =='':
            messagebox.showerror("category field is empty please fill now !")
        elif author =='':
            messagebox.showerror("author  field  is  empty please  fill  now  !")
        elif yop =='':
            messagebox.showerror("year of publishing field is empty please fill now !")
        else :
            try:
                self.c.execute('select book_name ,author ,year from library where book_name=%s
and category =%s and year =%s',(book_name,category,yop,))
                check = self.c.fetchone()
                if check ==None:
                    self.c.execute('insert  into  library(book_name,author,category,year)
values(%s,%s,%s,%s)',(book_name,author,category,yop,))
                    self.conn.commit()
                    self.show_library()
                    self.add.destroy()
                    messagebox.showinfo("Added !" ,"Selected Book is successfully Added Thank
you !")

                else:
                    messagebox.showinfo("already Added" ,"book is Already added")
                    self.add.destroy()
            except mysql.connector.Error as err:
                messagebox.showerror("Error", str(err))


    def add_window(self):
        self.deselect_all()
        self.add =Toplevel(self.root)
        self.add.geometry("400x400")
        self.add.title("Add new Book")
        self.add.resizable(False,False)
        self.add_window_frame = Frame(self.add)
        self.add_window_frame.config(bg='black')
        self.add_window_frame.pack(fill=BOTH , expand=TRUE)


        self.book_name =Label(self.add_window_frame,
                        text='Book name                              :', font=('Arial',10,'bold'),
                        fg='white',
                        bg='black')
        self.book_name_entry =Entry (self.add_window_frame,
                              font =("Arial")
                              )

        self.category =Label(self.add_window_frame,
                        text='Category                                :', font=('Arial',10,'bold'),
                        fg='white',
                        bg='black')
        self.category_entry =Entry (self.add_window_frame,
                              font =("Arial")
                              )

        self.author_name =Label(self.add_window_frame,
                        text='Author name                             :', font=('Arial',10,'bold'),
                        fg='white',
                        bg='black')
        self.author_name_entry =Entry (self.add_window_frame,
                              font =("Arial")
                              )
        self.yop =Label(self.add_window_frame,
                        text='published year :',
                        font=('Arial',10,'bold'),
                        fg='white',
                        bg='black')

        self.add_button =Button(self.add_window_frame ,
                          text ='Add',
                          font=("Comic Sans" ,13),
                          width=10,
                          command=self.add_to_library
                          )

        self.year_var = StringVar()
        self.year_selector = ttk.Combobox(self.add_window_frame, textvariable=self.year_var
,state='readonly')
        self.year_selector['values'] = tuple(range(1900, 2024))


        self.book_name.place(x=50 ,y=70)
        self.book_name_entry.place(x=160 ,y=70)
        self.category.place(x=50 ,y=120)
        self.category_entry.place(x=160 ,y=120)
        self.author_name.place(x=50 , y=170)
        self.author_name_entry.place(x=160 , y=170)
        self.yop.place(x=50 , y=220)
        self.year_selector.place(x=160 ,y=220)

        self.add_button.place(x=140 ,y=270)

    def create_side_bar_library_filter(self):

            self.c.execute('select count(book_name) from library where category =
%s',(self.library_select_category,))
            total_book = self.c.fetchone()

            total_category =self.library_select_category
            self.c.execute('select count(distinct(author)) from library where category =
%s',(self.library_select_category,))
            total_author = self.c.fetchone()

            self.total_book_library_filter_value.config(text = total_book)
            self.category_library_filter_value.config(text = total_category)

            self.author_library_filter_value.config(text = total_author)

    def create_side_bar_saved_filter(self):
        self.c.execute(' select count(sb.book_name) from saved_books sb join library l on
sb.product_id = l.product_id where l.category = %s and sb.id =
%s',(self.saved_select_category,self.id[0],))
        total_book = self.c.fetchone()
        self.c.execute(' select count(l.author) from saved_books sb join library l on sb.product_id
= l.product_id where l.category = %s and sb.id = %s',(self.saved_select_category,self.id[0],))
        total_author = self.c.fetchone()
        category = self.saved_select_category

        self.total_book_saved_filter_value.config(
                        text=total_book)
        self.category_saved_filter_value.config(
                        text=category)

        self.author_saved_filter_value  .config(
                        text=total_author)


    def  create_side_bar_library(self):
        self.c.execute('select count(book_name) from library')
        total_book = self.c.fetchone()
        self.c.execute('select  count(distinct(category))  from  library')
        total_category = self.c.fetchone()
        self.c.execute('select count(distinct(author)) from library')
        total_author = self.c.fetchone()

        self.total_book_library =Label(self.main_widget_frame,
                        text='Total Books                              :',
                        font=('Comic Sans',11,'bold'),
                        fg='white',
                        bg='black')
        self.category_library =Label(self.main_widget_frame,
                        text='Category ',
                        font=('Comic Sans',11,'bold'),
                        fg='white',
                        bg='black')
        self.author_library =Label(self.main_widget_frame,
                        text='author                                     :',
                        font=('Comic Sans',11,'bold'),
                        fg='white',
                        bg='black')

        self.total_book_library_value =Label(self.main_widget_frame,
                        text=total_book,
                        font=('Comic Sans',11,'bold'),
                        fg='white',
                        bg='black')
        self.category_library_value =Label(self.main_widget_frame,
                        text=total_category,
                        font=('Comic Sans',11,'bold'),
                        fg='white',
                        bg='black')
        self.author_library_value =Label(self.main_widget_frame,
                        text=total_author,
                        font=('Comic Sans',11,'bold'),
                        fg='white',
                        bg='black')

        if self.library_select_category ==  'all':
            self.total_book_library_filter_value  =Label(self.main_widget_frame,
                        text=total_book,
                        font=('Comic Sans',11,'bold'),
                        fg='white',
                        bg='black')
            self.category_library_filter_value =Label(self.main_widget_frame,
                        text='All',
                        font=('Comic Sans',8,'bold'),
                           fg='white',
                           bg='black')

                self.author_library_filter_value =  Label(self.main_widget_frame,
                           text=total_author,
                           font=('Comic Sans',11,'bold'),
                           fg='white',
                           bg='black')
            else :
                self.create_side_bar_library_filter()



        def  create_side_bar_saved(self):
            self.c.execute('select count(book_name) from saved_books where id = %s',self.id)
            total_book = self.c.fetchone()
            self.c.execute('select product_id from saved_books where id = %s',self.id)
            product_id = self.c.fetchall()

            flattened_values = tuple(value[0] for value in product_id)

            placeholders = ', '.join(['%s'] * len(flattened_values))
            query = f'SELECT count(distinct(category)) FROM library WHERE product_id IN
    ({placeholders})'

            self.c.execute(query, flattened_values)
            results = self.c.fetchone()
            category = results

            query = f'SELECT count(distinct(author)) FROM library WHERE product_id IN
    ({placeholders})'
            self.c.execute(query ,flattened_values)
            authors = self.c.fetchone()

            self.total_book_saved =Label(self.main_widget_frame,
                           text='Total Books                              :',
                           font=('Comic Sans',11,'bold'),
                           fg='white',
                           bg='black')
            self.category_saved =Label(self.main_widget_frame,
                           text='Category

                           :',
                           font=('Comic Sans',11,'bold'),
                           fg='white',
                           bg='black')
            self.author_saved =Label(self.main_widget_frame,
                           text='author                                   :',
                           font=('Comic Sans',11,'bold'),
                           fg='white',
                           bg='black')
            self.total_book_saved_value =Label(self.main_widget_frame,
                           text=total_book,
                           font=('Comic Sans',11,'bold'),
                           fg='white',
                           bg='black')
            self.category_saved_value =Label(self.main_widget_frame,
                           text=category,
    font=('Comic Sans',11,'bold'),
                            fg='white',
                            bg='black')
            self.author_saved_value  = Label(self.main_widget_frame,
                            text=authors,
                            font=('Comic Sans',11,'bold'),
                            fg='white',
                            bg='black')

            if  self.saved_select_category  ==  'all':
                self.total_book_saved_filter_value  =Label(self.main_widget_frame,
                            text=total_book,
                            font=('Comic Sans',11,'bold'),
                            fg='white',
                            bg='black')
                self.category_saved_filter_value =Label(self.main_widget_frame,
                            text = 'All',
                            font=('Comic Sans',8,'bold'),
                            fg='white',
                            bg='black')

                self.author_saved_filter_value = Label(self.main_widget_frame,
                            text=authors,
                            font=('Comic Sans',11,'bold'),
                            fg='white',
                            bg='black')
            else :
                self.create_side_bar_saved_filter()


        def  create_side_bar_common(self):
            underline_font = font.Font(family="Comic Sans", size=13, weight="bold",
    underline=True)
            self.total_book_sidebar =Label(self.main_widget_frame,
                            text='Total Books                               :',
                            font=('Comic Sans',11,'bold'),
                            fg='white',
                            bg='black')
            self.total_category_sidebar=Label(self.main_widget_frame,
                            text='Total category :',
                            font=('Comic Sans',11,'bold'),
                            fg='white',
                            bg='black')
            self.filter_sidebar=Label(self.main_widget_frame,
                            text='Filter',
                            font=underline_font,
                            fg='white',
                            bg='black')
            self.author_sidebar =Label(self.main_widget_frame,
                            text='author                                       :',
                            font=('Comic Sans',11,'bold'),
                            fg='white',
                            bg='black'
                            )

        def  create_saved_books(self):
            self.saved_book_page =Label(self.main_widget_frame,
                            text='saved books',
                           font=('Comic Sans',13,'bold'),
                           fg='white',
                           bg='black')
            self.delete_btn_saved =Button(self.main_widget_frame ,
                             text ='Delete',
                             font=("Comic Sans" ,13),
                             width=10,
                             command=self.delete_saved_listbox
                             )
            self.listbox_saved  =  Listbox(self.main_widget_frame,
                                  font=('Comic Sans', 12),
                                  width=45,
                                    height=15,
                                    bg ='#040404',
                                    fg='white')
            self.scrollbar_saved  =  Scrollbar(self.main_widget_frame,
                                       orient=VERTICAL,
                                       command=self.listbox_saved.yview,
                                       bg='gray')
            self.listbox_saved.config(yscrollcommand=self.scrollbar_saved.set ,
                                 selectmode=EXTENDED)
            self.listbox_saved.bind('<Double-Button-1>',  self.on_double_click)


            self.search_saved_entry = Entry(  self.main_widget_frame  ,font  =  "Arial")
            self.search_saved_button = Button(self.main_widget_frame ,
                             text ='Search',
                             font=("Comic Sans" ,11),
                             width=10,
                             command=self.saved_search_command
                             )


        def  create_change_password(self):
            self.change_password_page =Label(self.main_widget_frame,
                           text='Change Password',
                           font=('Comic Sans',15,'bold'),
                           fg='white',
                           bg='black')

            self.old_password =Label(self.main_widget_frame,
                           text='Enter old password :',
                           font=('Arial',10,'bold'),
                           fg='white',
                           bg='black')
            self.new_password =Label(self.main_widget_frame,
                           text='Enter new password :',
                           font=('Arial',10,'bold'),
                           fg='white',
                           bg='black')
            self.old_password_entry  =  Entry (self.main_widget_frame,
                                      font =("Arial")
                                      )
            self.new_password_entry  =  Entry (self.main_widget_frame,
                                      font =("Arial"),
                                      show="*"
                                      )
            self.change_password_btn =Button(self.main_widget_frame ,
                            text ='Change',
                            font=("Comic Sans" ,13),
                            width=12,
                            command=self.change_password

                            )



        def show_library(self):
            self.hide_main_frame()
            self.create_main_widget()
            self.insert_listbox()
            self.insert_to_category()
            self.create_side_bar_common()
            self.create_side_bar_library()

            self.library_page.place(x=5 ,y=55)
            self.search_library_entry.place(x=80 ,y=56,width=280)
            self.search_library_button.place(x=365,y=56,width =50  ,height=23)
            self.book_listbox.place(x=5, y=80)

            self.list_scrollbar.place(x=396, y=80, height=289)
            self.add_btn.place(x=30,y=380)
            self.save_btn.place(x=160 ,y=380)
            self.delete_btn.place(x=290 ,y=380)
            self.total_book_sidebar.place( x= 430 , y=80)
            self.total_category_sidebar.place(x=430 ,y =120)
            self.author_sidebar.place(x=430 ,y=160)
            self.filter_sidebar.place(x=485 ,y=200)
            self.total_book_library.place(x=430 ,y=240)
            self.category_library.place(x=430 ,y=320)
            self.author_library.place(x=430 ,y=280)

            self.total_book_library_filter_value.place(x=550 ,y=240)
            self.category_library_filter_value.place(x=430 ,y=360)
            self.author_library_filter_value.place(x=550 ,y=280)


            self.total_book_library_value.place(x=550   ,y=80)
            self.category_library_value.place(x=550  ,y  =120)
            self.author_library_value.place(x=550,y=160)

            window.geometry("600x420")


        def show_saved_books(self):
            self.hide_main_frame()
            self.create_main_widget()
            self.insert_saved_listbox()
            self.insert_to_saved()
            self.create_side_bar_common()
            self.create_side_bar_saved()


            self.saved_book_page.place(x=5  ,y=55)
            self.search_saved_entry.place(x=110 ,y=56,width=250)
            self.search_saved_button.place(x=365,y=56,width =50 ,height=23)
            self.listbox_saved.place(x=5, y=80)
            self.scrollbar_saved.place(x=396, y=80, height=289)


            self.delete_btn_saved.place(x=160 ,y=380)
            self.total_book_sidebar.place( x= 430 , y=80)
            self.total_category_sidebar.place(x=430 ,y =120)
            self.author_sidebar.place(x=430 ,y=160)
            self.filter_sidebar.place(x=485 ,y=200)
            self.total_book_saved.place(x=430 ,y=240)
            self.category_saved.place(x=430 ,y=320)
            self.author_saved.place(x=430 ,y=280)

            self.total_book_saved_filter_value.place(x=550 ,y=240)
            self.category_saved_filter_value.place(x=430,y=360)
            self.author_saved_filter_value.place(x=550 ,y=280)

            self.total_book_saved_value.place(x=550 , y=80)
            self.category_saved_value.place(x=550 ,y=120)
            self.author_saved_value.place(x=550 ,y =160)


            window.geometry("600x420")


        def show_change_password(self):
            self.hide_main_frame()
            self.create_main_widget()
            self.menu_bar.entryconfig(2, state="disabled")

            self.change_password_page.place(x=120,y=80)
            self.old_password.place(x=50 ,y=150)
            self.old_password_entry.place(x=200 ,y=150)
            self.new_password.place(x=50 ,y=200)
            self.new_password_entry.place(x=200 ,y=200)
            self.change_password_btn.place(x =200 ,y =250)
            window.geometry("420x420")


        def create_profile(self):
            self.c.execute('select name, username from user where id = %s', self.id)
            name ,username = self.c.fetchone()
            self.c.execute('select count(id) from saved_books where id =%s',self.id)
            saved_books = self.c.fetchone()

            self.profile_page =Label(self.main_widget_frame,
                            text='Profile',
                            font=('Comic Sans',15,'bold'),
                            fg='white',
                            bg='black')

            self.show_name =Label(self.main_widget_frame,
                            text='Name                                    :', font=('Arial',10,'bold'),
                            fg='white',
                            bg='black')
            self.show_username =Label(self.main_widget_frame,
                            text='UserName    :',
                            font=('Arial',10,'bold'),
                            fg='white',
                            bg='black')
            self.show_savedbooks =Label(self.main_widget_frame,
                            text='Books                                    :',
                            font=('Arial',10,'bold'),
                            fg='white',
                            bg='black')

            self.show_name_value =Label(self.main_widget_frame,
                            text=name,
                            font=('Arial',10,'bold'),
                            fg='white',
                            bg='black')

            self.show_username_value =Label(self.main_widget_frame,
                            text=username,
                            font=('Arial',10,'bold'),
                            fg='white',
                            bg='black')
            self.show_savedbooks_value =Label(self.main_widget_frame,
                            text =saved_books[0],
                            font=('Arial',10,'bold'),
                            fg='white',
                            bg='black')

        def show_profile(self):
            self.hide_main_frame()

            self.create_main_widget()

            #hide menu bar
            self.menu_bar.entryconfig(2,  state="disabled")


            self.profile_page.place(x=150 ,y=80)
            self.show_name_value.place(x=150,y=130)
            self.show_username_value.place(x=150,y=170)
            self.show_name.place(x=50,y=130)
            self.show_username.place(x=50,y=170)
            self.show_savedbooks.place(x=50,y=210)
            self.show_savedbooks_value.place(x=150,y=210)
            window.geometry("420x420")

    window = Tk()

    app = ReadEase(window)
    window.mainloop()
```

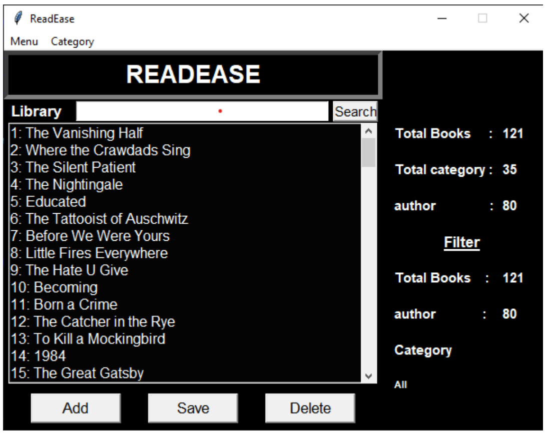
5. RESULTS





Welcome Sihabutheen ,

wait for 5 seconds, it will automatically redirect library page

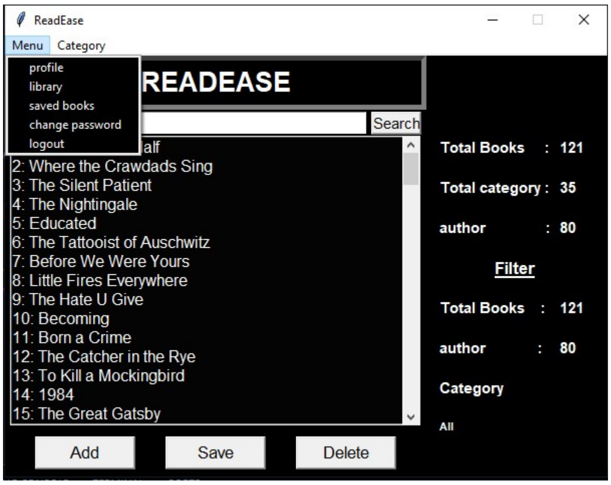


1: The Vanishing Half
2: Where the Crawdads Sing
3: The Silent Patient
4: The Nightingale
5: Educated
6: The Tattooist of Auschwitz
7: Before We Were Yours
8: Little Fires Everywhere
9: The Hate U Give
10: Becoming
11: Born a Crime
12: The Catcher in the Rye
13: To Kill a Mockingbird
14: 1984
15: The Great Gatsby

Total Books : 121
Total category : 35
author : 80

Filter

Total Books : 121
author : 80
Category
All



READEASE

alf
2: Where the Crawdads Sing
3: The Silent Patient
4: The Nightingale
5: Educated
6: The Tattooist of Auschwitz
7: Before We Were Yours
8: Little Fires Everywhere
9: The Hate U Give
10: Becoming
11: Born a Crime
12: The Catcher in the Rye
13: To Kill a Mockingbird
14: 1984
15: The Great Gatsby

Total Books : 121

Total category : 35

author : 80

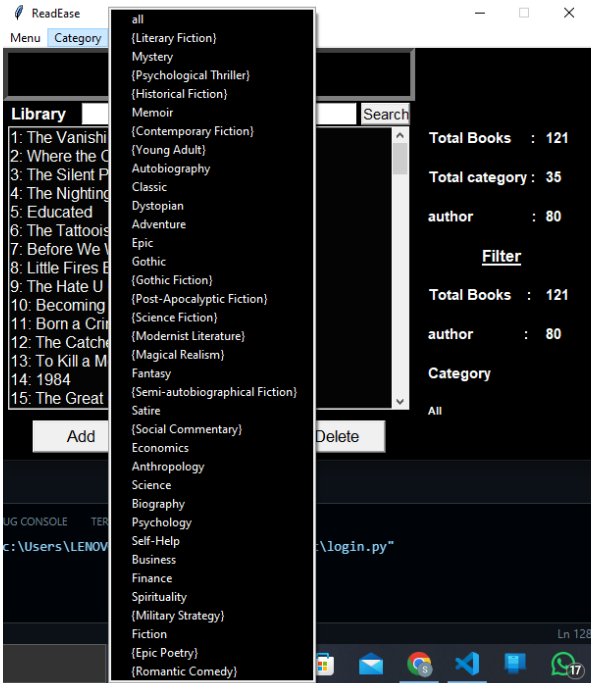
Filter

Total Books : 121

author : 80

Category

All



Library

1: The Vanishi
2: Where the C
3: The Silent P
4: The Nighting
5: Educated
6: The Tattoois
7: Before We V
8: Little Fires E
9: The Hate U
10: Becoming
11: Born a Cri
12: The Catche
13: To Kill a M
14: 1984
15: The Great

Total Books : 121

Total category : 35

author : 80

Filter

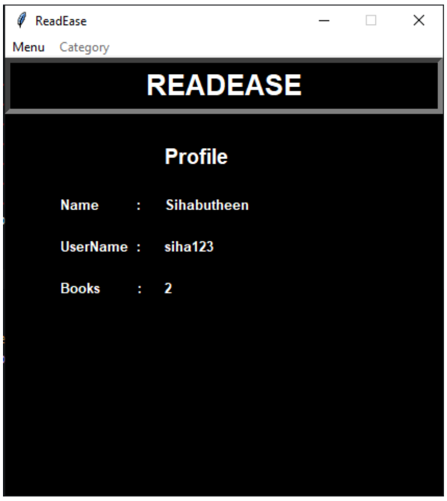
Total Books : 121

author : 80

Category

All

```
c:\Users\LENOV                    :\login.py"
```



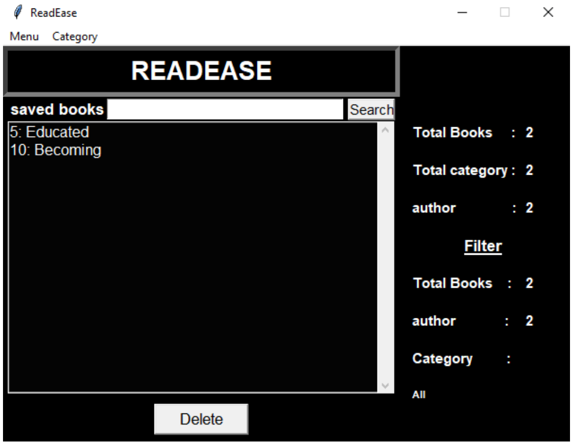
READEASE

Profile

Name : Sihabutheen

UserName : siha123

Books : 2



READEASE

saved books

5: Educated
10: Becoming

Total Books : 2

Total category : 2

author : 2

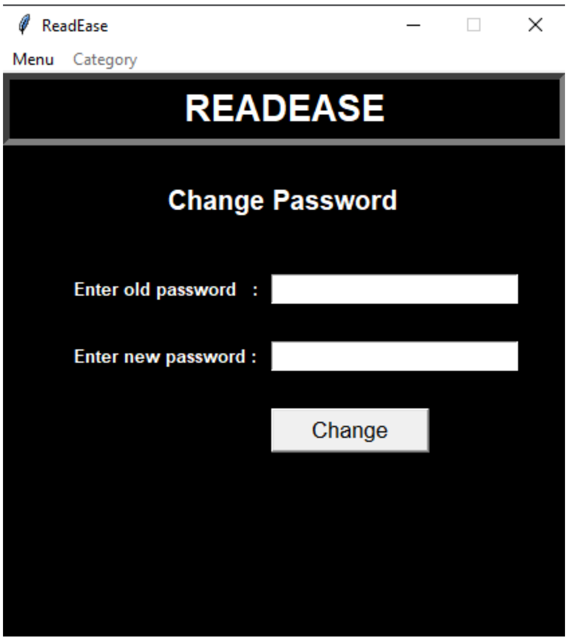
Filter

Total Books : 2

author : 2

Category :

All

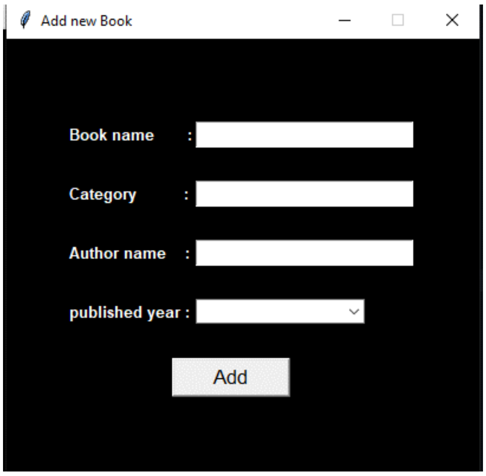


READEASE

Change Password

Enter old password :

Enter new password :

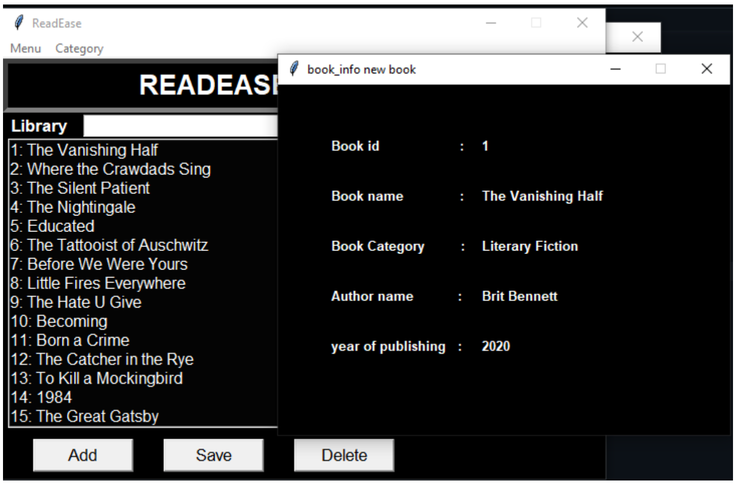


Book name :

Category :

Author name :

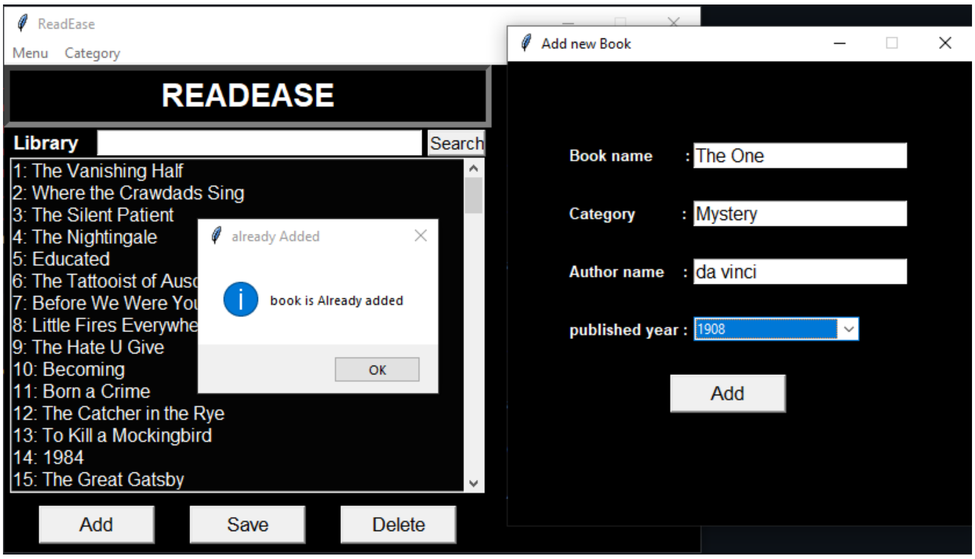
published year :



1: The Vanishing Half
2: Where the Crawdads Sing
3: The Silent Patient
4: The Nightingale
5: Educated
6: The Tattooist of Auschwitz
7: Before We Were Yours
8: Little Fires Everywhere
9: The Hate U Give
10: Becoming
11: Born a Crime
12: The Catcher in the Rye
13: To Kill a Mockingbird
14: 1984
15: The Great Gatsby

book_info new book

Book id	:	1
Book name	:	The Vanishing Half
Book Category	:	Literary Fiction
Author name	:	Brit Bennett
year of publishing	:	2020



1: The Vanishing Half
2: Where the Crawdads Sing
3: The Silent Patient
4: The Nightingale
5: Educated
6: The Tattooist of Aus
7: Before We Were You
8: Little Fires Everywhe
9: The Hate U Give
10: Becoming
11: Born a Crime
12: The Catcher in the Rye
13: To Kill a Mockingbird
14: 1984
15: The Great Gatsby

already Added

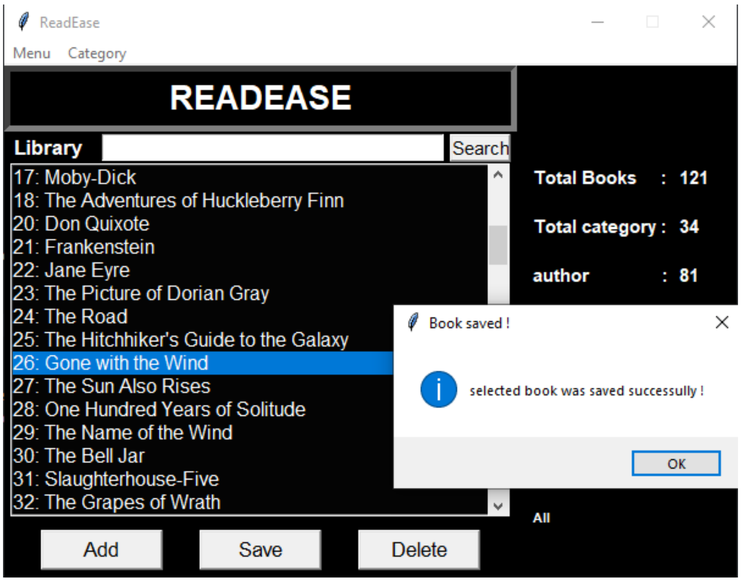
book is Already added

OK

Add new Book

Book name	:	The One
Category	:	Mystery
Author name	:	da vinci
published year	:	1908

Add



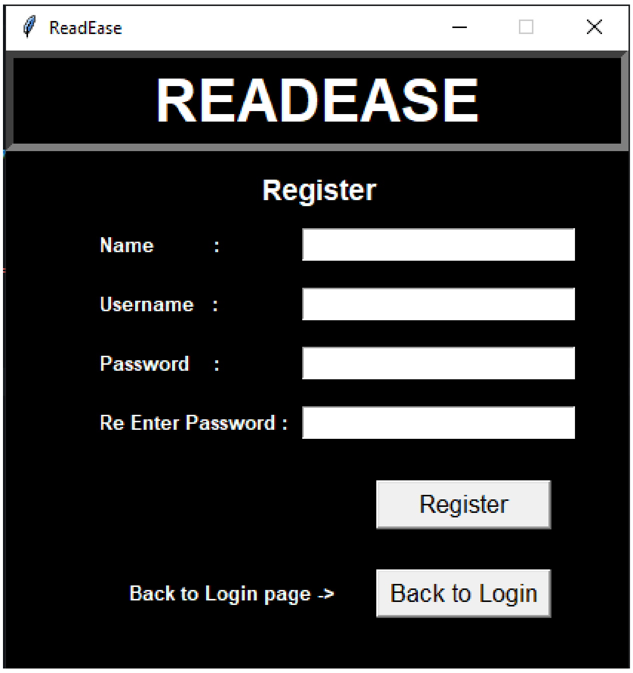
17: Moby-Dick
18: The Adventures of Huckleberry Finn
20: Don Quixote
21: Frankenstein
22: Jane Eyre
23: The Picture of Dorian Gray
24: The Road
25: The Hitchhiker's Guide to the Galaxy
26: Gone with the Wind
27: The Sun Also Rises
28: One Hundred Years of Solitude
29: The Name of the Wind
30: The Bell Jar
31: Slaughterhouse-Five
32: The Grapes of Wrath

Total Books	:	121
Total category	:	34
author	:	81

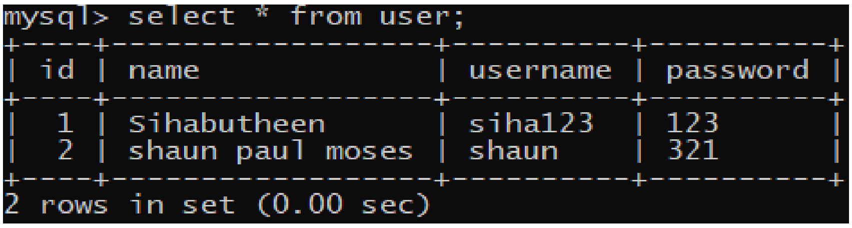
Book saved !

selected book was saved successully !

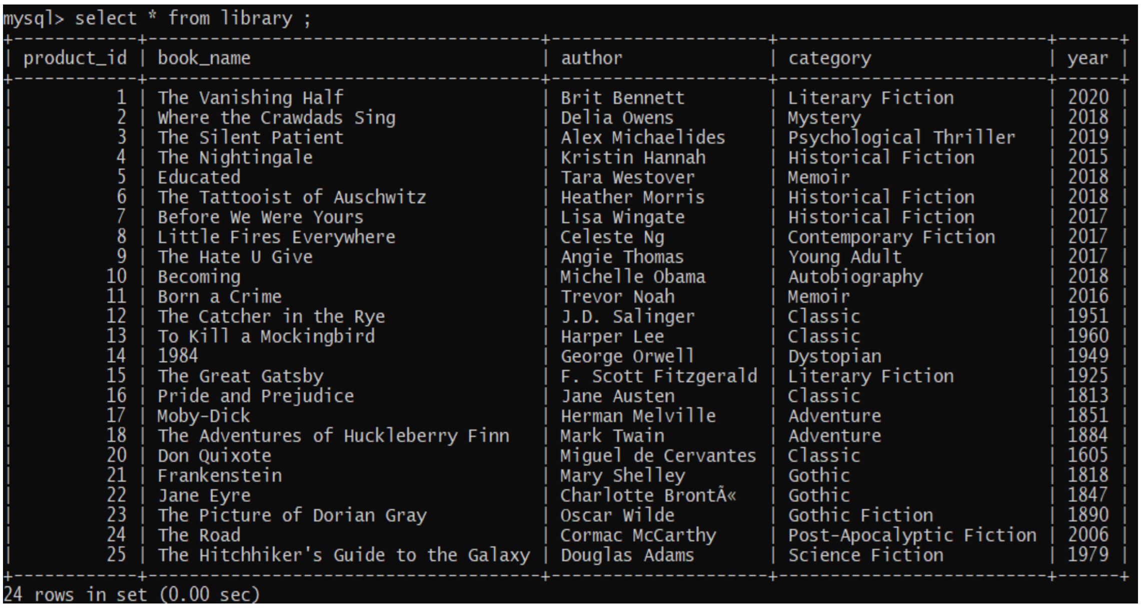
OK



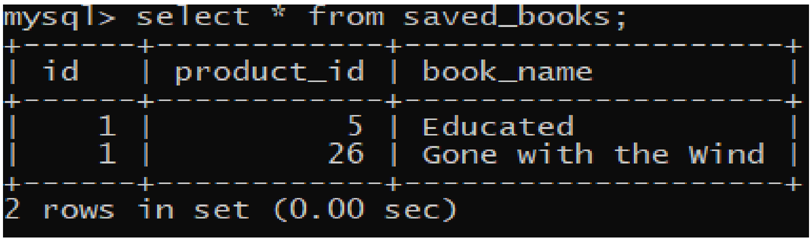
DATABASE – MYSQL :



```
mysql> select * from user;
+----+--------------------+----------+----------+
| id | name               | username | password |
+----+--------------------+----------+----------+
|  1 | Sihabutheen        | siha123  | 123      |
|  2 | shaun paul moses   | shaun    | 321      |
+----+--------------------+----------+----------+
2 rows in set (0.00 sec)
```



```
mysql> select * from library ;
+------------+--------------------------------------+---------------------+--------------------------+------+
| product_id | book_name                            | author              | category                 | year |
+------------+--------------------------------------+---------------------+--------------------------+------+
|          1 | The Vanishing Half                   | Brit Bennett        | Literary Fiction         | 2020 |
|          2 | Where the Crawdads Sing              | Delia Owens         | Mystery                  | 2018 |
|          3 | The Silent Patient                   | Alex Michaelides    | Psychological Thriller   | 2019 |
|          4 | The Nightingale                      | Kristin Hannah      | Historical Fiction       | 2015 |
|          5 | Educated                             | Tara Westover       | Memoir                   | 2018 |
|          6 | The Tattooist of Auschwitz           | Heather Morris      | Historical Fiction       | 2018 |
|          7 | Before We Were Yours                 | Lisa Wingate        | Historical Fiction       | 2017 |
|          8 | Little Fires Everywhere              | Celeste Ng          | Contemporary Fiction     | 2017 |
|          9 | The Hate U Give                      | Angie Thomas        | Young Adult              | 2017 |
|         10 | Becoming                             | Michelle Obama      | Autobiography            | 2018 |
|         11 | Born a Crime                         | Trevor Noah         | Memoir                   | 2016 |
|         12 | The Catcher in the Rye               | J.D. Salinger       | Classic                  | 1951 |
|         13 | To Kill a Mockingbird                | Harper Lee          | Classic                  | 1960 |
|         14 | 1984                                 | George Orwell       | Dystopian                | 1949 |
|         15 | The Great Gatsby                     | F. Scott Fitzgerald | Literary Fiction         | 1925 |
|         16 | Pride and Prejudice                  | Jane Austen         | Classic                  | 1813 |
|         17 | Moby-Dick                            | Herman Melville     | Adventure                | 1851 |
|         18 | The Adventures of Huckleberry Finn   | Mark Twain          | Adventure                | 1884 |
|         20 | Don Quixote                          | Miguel de Cervantes | Classic                  | 1605 |
|         21 | Frankenstein                         | Mary Shelley        | Gothic                   | 1818 |
|         22 | Jane Eyre                            | Charlotte BrontÃ«    | Gothic                   | 1847 |
|         23 | The Picture of Dorian Gray           | Oscar Wilde         | Gothic Fiction           | 1890 |
|         24 | The Road                             | Cormac McCarthy     | Post-Apocalyptic Fiction | 2006 |
|         25 | The Hitchhiker's Guide to the Galaxy | Douglas Adams       | Science Fiction          | 1979 |
+------------+--------------------------------------+---------------------+--------------------------+------+
24 rows in set (0.00 sec)
```



```
mysql> select * from saved_books;
+------+------------+---------------------+
| id   | product_id | book_name           |
+------+------------+---------------------+
|    1 |          5 | Educated            |
|    1 |         26 | Gone with the Wind  |
+------+------------+---------------------+
2 rows in set (0.00 sec)
```

CHAPTER -6

CONCLUSIONS

Readease, our online library management system, represents a modern solution for managing library resources with ease and accessibility. By harnessing the power of digital technology, users can seamlessly add, delete, and save books to the library from anywhere, at any time. Readease empowers librarians and users alike by offering a streamlined platform for organizing and accessing literary resources. The integration of features such as online book management and real-time updates ensures efficiency and convenience in library operations.With Readease, the traditional constraints of physical libraries are transcended, opening up a world of literary exploration and knowledge dissemination. As we continue to innovate and improve, Readease remains committed to enhancing the reading experience for all users, fostering a culture of lifelong learning and discovery

CHAPTER – 7

REFERENCES

1 . Manish Kumar Srivastava, A.K Tiwari, “A Study of Behavior of Maruti SX4 and Honda City Custo Jaipur”, Pacific Business Review- Quarterly Referred Journal, Zenith International Journal of Multi disciplinary Research Vol.4, Issue 4, pp. 77-90, Apr2011.

2. M.Prasanna Mohan Raj, Jishnu Sasikumar, S.Sriram , “A Study of Customers Brand Preference inSUVS and MUVS: Effect on Marketing Mix Variables”, International Referred Research Journal Vol.- IV, Issue-1, pp. 48-58, Jan2013.

3. Nikhil Monga, Bhuvender Chaudhary, “Car Market and Buying behavior - study on ConsumerPerception”, IJRMEC Vol.2, Issue-2, pp. 44-63, Feb2012 .

❿ Tkinter Documentation: Link to Tkinter Documentation

❿ MySQL Documentation: Link to MySQL Documentation

❿ Python Documentation: Link to Python Documentation