



DELHI PUBLIC SCHOOL BANGALORE NORTH
COMPUTER SCIENCE PROJECT REPORT
(2024-25)

Team Member Name	Class/ Sec	Adm. No.	Board Roll No.
Anuj Krish Nair	XII C	BN/5122	18608266
Sriharsha Mallavarapu	XII C	BN/19631	18608283
Adwait Hemanth	XII A	BN/19638	18608286

ACKNOWLEDGEMENT

This project was possible only because of the help, advice, and hard work of our supportive teachers. We would like to show our gratitude to the principal, Mrs. Manju Balasubramanyam, for giving us the opportunity and means to work on this project. We are also very grateful to our Computer Teachers, Mrs. Uzma Fathima, and Mrs. Manjula S for encouraging us and providing the required guidance. We would like to thank our friends, classmates, and our parents for their valuable input into this project.

TABLE OF CONTENTS

1. Introduction
 - Purpose
 - Motivation
 - Tools and Technologies
2. Hardware and Software Specifications
 - Development Platform
 - Deployment Platform
3. Data Collection and Preprocessing
 - Book Data
 - User Preferences
4. Project System Design
5. Project Timeline Chart
6. Implementation
7. Testing
8. Source Code
9. Project Snapshots
10. Conclusion
11. Further Enhancements
12. Bibliography

1. Introduction:

Purpose

This report outlines the development and implementation of a Library Kiosk System. The purpose of this project is to simplify library operations by providing an intuitive system for users to borrow and return books, along with a user-friendly login interface.

Motivation

The motivation behind this project stems from the need to modernize traditional library systems, making them more accessible and efficient. By leveraging Python and Tkinter, this project provides an easy-to-use platform for both library users and administrators.

Tools and Technologies

- **Programming Language:** Python
- **Frameworks:** Tkinter
- **Libraries:** Pillow (for image processing), JSON (for data storage)
- **Others:** urllib (for downloading book images)

2. Hardware and Software Specifications:

Development Platform

- **Operating System:** Windows 10/11 or Linux
- **IDE/Editor:** PyCharm, VS Code
- **Hardware:**

- Processor: Intel Core i5 or equivalent
- RAM: 8 GB or more
- Storage: 256 GB SSD

Deployment Platform

- **Deployment Environment:** Local system
- **Requirements:** Python runtime

3. Data Collection and Preprocessing:

Book Data

Data for the library kiosk system was manually curated and stored in a JSON file. Each book entry includes:

- Title
- Author
- Availability status
- Image URL

User Preferences

User preferences and credentials were stored in a separate JSON file, including:

- Username
- Password

4. Project System Design:

The system design follows a modular approach, including:

1. **User Interface Module:** Built with Tkinter for interaction.
2. **Data Management Module:** Handles the loading, saving, and modification of book and user data.
3. **Image Handling Module:** Integrates book cover images using the Pillow library.

System Design Flowchart

Library Kiosk Flowchart

Start -> Login Screen -> [Login Successful?]

[Yes]: Go to Main Menu

```
|
|---> View Books -> Books List Screen -> Back to Main Menu
|
|---> Borrow Book -> Borrow Book Flow -> Back to Main Menu
|
|---> Return Book -> Return Book Flow -> Back to Main Menu
|
|---> Logout -> Back to Login Screen
```

[No]: Display Error Message -> Back to Login Screen

End

5. Project Timeline Chart:

Task	Start Date	End Date	Status
Data Collection	01-01-2025	07-01-2025	Completed
System Design	08-01-2025	15-01-2025	Completed
Implementation	16-01-2025	31-01-2025	Ongoing
Testing	01-02-2025	07-02-2025	Upcoming
Deployment	08-02-2025	15-02-2025	Pending

6. Implementation:

The project was implemented in the following phases:

1. **Phase 1:** Development of the user login system.
2. **Phase 2:** Integration of book data with JSON.
3. **Phase 3:** Building the borrowing and returning module.
4. **Phase 4:** Enhancing UI with book images using Pillow.

Key Features Implemented:

- User login system with validation.
- Book borrowing and returning functionality.
- Dynamic book cover display using URLs and local caching.

7. Testing:

Testing was carried out in two stages:

1. **Unit Testing:** Each module was tested individually to ensure functionality.
2. **Integration Testing:** End-to-end testing to ensure seamless operation.

Test Case	Input	Expected Output	Result
Login Validation	Valid creds	Successful login	Passed
Book Viewing	Nil	All books displayed	Passed
Book Borrowing	Book title	Book marked as borrowed	Passed
Book Returning	Book title	Book marked as available	Passed
Logout	Nil	Return to login screen	Passed

8. Source Code:

The full source code for the Library Kiosk System is as follows:

```
import tkinter as tk # GUI framework for building the application
from tkinter import messagebox, simpledialog, ttk # Widgets for user interaction
from PIL import Image, ImageTk # For handling images in the GUI
import json # For reading and writing JSON files
import os # For file handling
import urllib.request # For downloading images from URLs

# JSON Database Files
user_db_file = "user_db.json" # Path to the user database file
book_db_file = "book_db.json" # Path to the book database file

# Initial Database Structure
def initialize_database():
    """Initialize the database with default users and books if not already present."""
    # Check and create user database
    if not os.path.exists(user_db_file):
        users = {
            "users": {
                "admin": {"password": "admin123"},
                "user1": {"password": "password1"}
            }
        }
        with open(user_db_file, 'w') as f:
            json.dump(users, f, indent=4)

    # Check and create book database
    if not os.path.exists(book_db_file):
        books = {
            "books": [
                {"title": "1984", "author": "George Orwell", "is_available": True,
                 "image": "https://covers.openlibrary.org/b/id/7222246-L.jpg"},
                {"title": "To Kill a Mockingbird", "author": "Harper Lee", "is_available": True,
                 "image": "https://covers.openlibrary.org/b/id/8228691-L.jpg"},
                {"title": "The Great Gatsby", "author": "F. Scott Fitzgerald", "is_available": True,
                 "image": "https://covers.openlibrary.org/b/id/6516725-L.jpg"}
            ]
        }
        with open(book_db_file, 'w') as f:
            json.dump(books, f, indent=4)
```

Load JSON Database

```
def load_database(file):  
    """Load the JSON database from the given file."""  
    with open(file, 'r') as f:  
        return json.load(f)
```

Save JSON Database

```
def save_database(data, file):  
    """Save the given data to the specified JSON file."""  
    with open(file, 'w') as f:  
        json.dump(data, f, indent=4)
```

Main Library Kiosk Class

```
class LibraryKiosk:  
    """Main class for the library kiosk system."""  
  
    def __init__(self, root):  
        """Initialize the Library Kiosk GUI and set up the login screen."""  
        if hasattr(self, 'books_frame'): # Check if books_frame exists  
            self.books_frame.destroy() # Destroy books_frame if it exists  
        self.root = root  
        self.root.title("Library Kiosk System")  
        self.current_user = None # Keep track of the currently logged-in user  
  
        # Login Frame  
        self.login_frame = tk.Frame(root)  
        self.login_frame.pack(pady=50)  
  
        tk.Label(self.login_frame, text="Library Kiosk Login", font=("Helvetica", 16)).pack(pady=10)  
        tk.Label(self.login_frame, text="Username:").pack()  
        self.username_entry = tk.Entry(self.login_frame) # Entry field for username  
        self.username_entry.pack()  
  
        tk.Label(self.login_frame, text="Password:").pack()  
        self.password_entry = tk.Entry(self.login_frame, show="*") # Entry field for password (masked)  
        self.password_entry.pack()  
  
        tk.Button(self.login_frame, text="Login", command=self.login).pack(pady=10) # Button to initiate  
login
```

```

def login(self):
    """Handle user login and validation."""
    username = self.username_entry.get()
    password = self.password_entry.get()

    db = load_database(user_db_file) # Load user database

    # Validate username and password
    if username in db["users"] and db["users"][username]["password"] == password:
        self.current_user = username # Set the current user
        messagebox.showinfo("Success", f"Welcome, {username}!")
        self.show_main_menu() # Show the main menu
    else:
        messagebox.showerror("Error", "Invalid username or password")

def show_main_menu(self):
    """Display the main menu after successful login."""
    if hasattr(self, 'login_frame'): # Check if login_frame exists
        self.login_frame.destroy() # Destroy the login frame
    if hasattr(self, 'books_frame'): # Check if books_frame exists
        self.books_frame.destroy() # Destroy books_frame if it exists

    self.main_frame = tk.Frame(self.root)
    self.main_frame.pack(pady=50)

    tk.Label(self.main_frame, text=f"Welcome, {self.current_user}!", font=("Helvetica",
16)).pack(pady=10)
    tk.Button(self.main_frame, text="View Books", command=self.view_books).pack(pady=5)
    tk.Button(self.main_frame, text="Borrow Book", command=self.borrow_book).pack(pady=5)
    tk.Button(self.main_frame, text="Return Book", command=self.return_book).pack(pady=5)
    tk.Button(self.main_frame, text="Logout", command=self.logout).pack(pady=5)

def view_books(self):
    """Display a list of books with their details and availability."""
    if hasattr(self, 'books_frame'): # Check if books_frame exists
        self.books_frame.destroy() # Destroy books_frame if it exists
    db = load_database(book_db_file) # Load book database
    books = db["books"]

    self.main_frame.destroy() # Remove the main menu frame
    self.books_frame = tk.Frame(self.root)
    self.books_frame.pack(pady=20)

    tk.Label(self.books_frame, text="Library Books", font=("Helvetica", 16)).pack(pady=10)

```

```

# Create a scrollable frame for displaying books
canvas = tk.Canvas(self.books_frame)
scrollbar = ttk.Scrollbar(self.books_frame, orient="vertical", command=canvas.yview)
scrollable_frame = tk.Frame(canvas)

scrollable_frame.bind(
    "<Configure>",
    lambda e: canvas.configure(
        scrollregion=canvas.bbox("all")
    )
)

canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
canvas.configure(yscrollcommand=scrollbar.set)

# Loop through books and display their details with an image
for book in books:
    frame = tk.Frame(scrollable_frame, borderwidth=2, relief="groove")
    frame.pack(pady=10, padx=10, fill="x")

    try:
        # Download and display book image
        image_url = book.get("image", "")
        image_file = f"temp_{book['title'].replace(' ', '_')}.jpg"
        urllib.request.urlretrieve(image_url, image_file)
        img = Image.open(image_file)
        img = img.resize((100, 150)) # Resize the image
        photo = ImageTk.PhotoImage(img)
    except Exception as e:
        photo = None # Fallback if the image fails to load

    if photo:
        image_label = tk.Label(frame, image=photo)
        image_label.image = photo
        image_label.pack(side="left", padx=10)

    # Display book details
    text = f"Title: {book['title']}\nAuthor: {book['author']}\nStatus: {'Available' if book['is_available']\nelse 'Borrowed'}"
    tk.Label(frame, text=text, justify="left", anchor="w").pack(side="left", padx=10)

canvas.pack(side="left", fill="both", expand=True)
scrollbar.pack(side="right", fill="y")

```

```
tk.Button(self.books_frame, text="Back", command=self.show_main_menu).pack(pady=10)
```

```
def borrow_book(self):
```

```
    """Allow the user to borrow an available book."""
```

```
    if hasattr(self, 'books_frame'): # Check if books_frame exists
```

```
        self.books_frame.destroy() # Destroy books_frame if it exists
```

```
    db = load_database(book_db_file)
```

```
    books = db["books"]
```

```
    # Filter for available books
```

```
    available_books = [book for book in books if book["is_available"]]
```

```
    if not available_books:
```

```
        messagebox.showinfo("No Books", "No books are currently available for borrowing.")
```

```
        return
```

```
    book_titles = [book['title'] for book in available_books]
```

```
    # Ask the user to enter the title of the book they want to borrow
```

```
    book_to_borrow = simpledialog.askstring("Borrow Book",
```

```
        f"Available Books:\n{chr(10).join(book_titles)}\n\nEnter book title to
```

```
borrow:")
```

```
    if book_to_borrow:
```

```
        for book in books:
```

```
            if book['title'].lower() == book_to_borrow.lower() and book["is_available"]:
```

```
                book["is_available"] = False # Mark the book as borrowed
```

```
                save_database({"books": books}, book_db_file)
```

```
                messagebox.showinfo("Success", f"You borrowed '{book['title']}'!")
```

```
                return
```

```
        messagebox.showerror("Error", "Book not found or already borrowed.")
```

```
def return_book(self):
```

```
    """Allow the user to return a borrowed book."""
```

```
    if hasattr(self, 'books_frame'): # Check if books_frame exists
```

```
        self.books_frame.destroy() # Destroy books_frame if it exists
```

```
    db = load_database(book_db_file)
```

```
    books = db["books"]
```

```
    # Filter for borrowed books
```

```
    borrowed_books = [book for book in books if not book["is_available"]]
```

```
    if not borrowed_books:
```

```
        messagebox.showinfo("No Books", "No books are currently borrowed.")
```

```
        return
```

```

book_titles = [book['title'] for book in borrowed_books]
# Ask the user to enter the title of the book they want to return
book_to_return = simpledialog.askstring("Return Book",
                                         f"Borrowed Books:\n{chr(10).join(book_titles)}\n\nEnter book title to
return:")

```

```

if book_to_return:
    for book in books:
        if book['title'].lower() == book_to_return.lower() and not book["is_available"]:
            book["is_available"] = True # Mark the book as available
            save_database({"books": books}, book_db_file)
            messagebox.showinfo("Success", f"You returned '{book['title']}'!")
        return
    messagebox.showerror("Error", "Book not found or already available.")

```

```

def logout(self):
    """Logout the current user and return to the login screen."""
    self.books_frame.destroy()
    if hasattr(self, 'main_frame'): # Check if main_frame exists
        self.main_frame.destroy() # Destroy the main menu frame
    self.__init__(self.root)

```

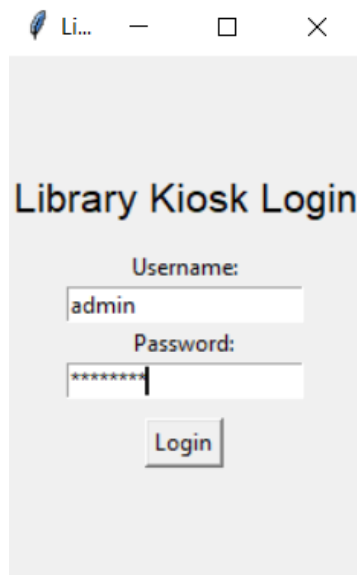
Main Function

```

if __name__ == "__main__":
    initialize_database() # Set up the initial database
    root = tk.Tk()
    app = LibraryKiosk(root) # Initialize the Library Kiosk application
    root.mainloop() # Start the Tkinter event loop

```

9. Project Snapshots:

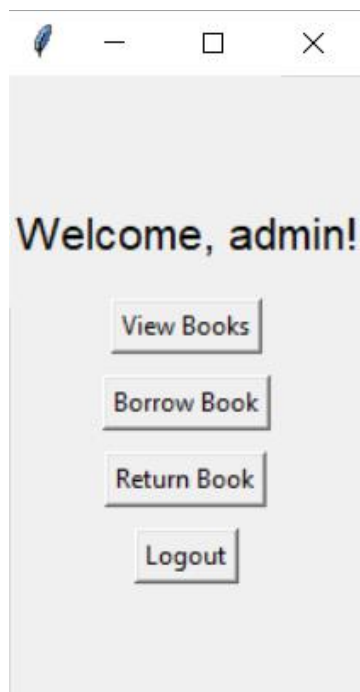


Library Kiosk Login

Username:

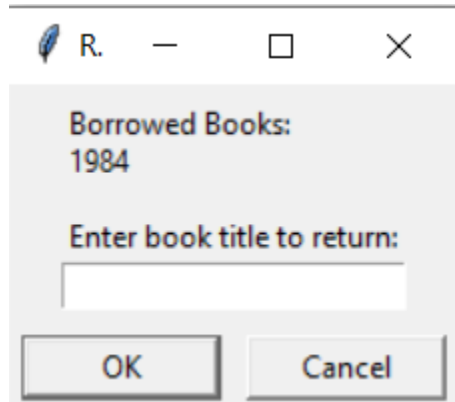
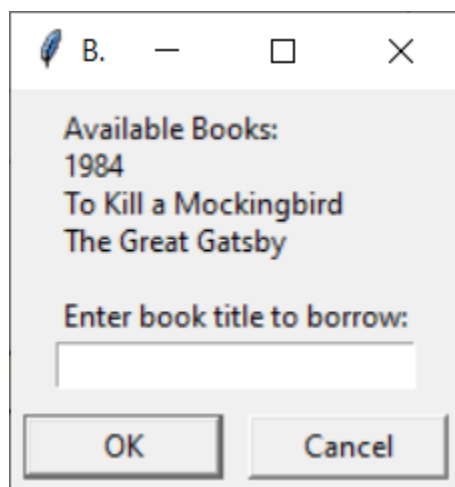
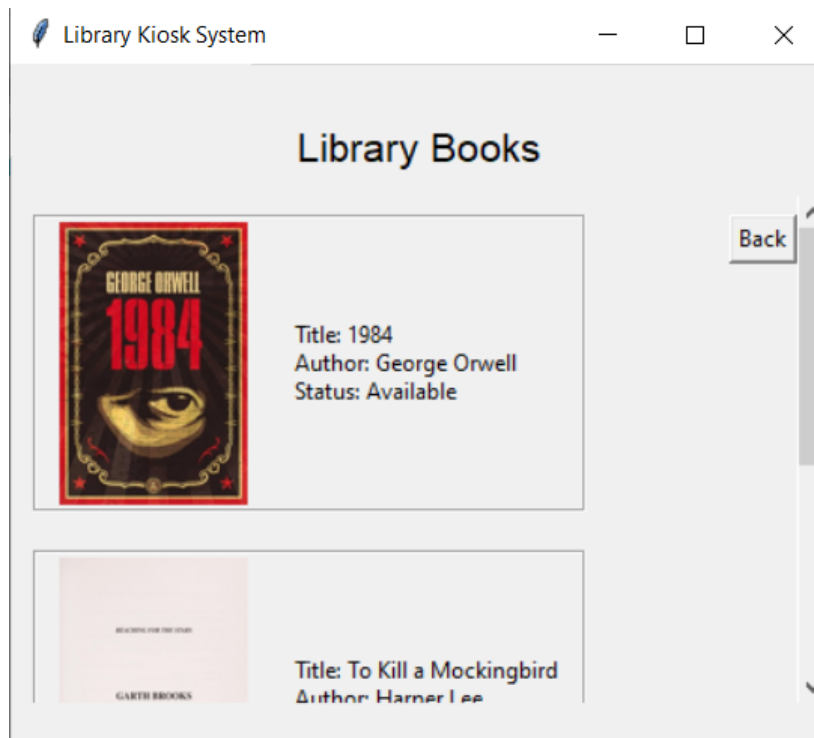
Password:

This is a screenshot of a web browser window titled 'Library Kiosk Login'. The window has a standard title bar with a feather icon, a minus sign, a square icon, and a close button. The main content area is light gray and contains the title 'Library Kiosk Login' in a large, bold, black font. Below the title, there are two input fields: 'Username:' with the text 'admin' and 'Password:' with masked characters '*****'. A 'Login' button is positioned below the password field.



Welcome, admin!

This is a screenshot of a web browser window titled 'Welcome, admin!'. The window has a standard title bar with a feather icon, a minus sign, a square icon, and a close button. The main content area is light gray and contains the title 'Welcome, admin!' in a large, bold, black font. Below the title, there are four buttons arranged vertically: 'View Books', 'Borrow Book', 'Return Book', and 'Logout'.



10. Conclusion:

The Library Kiosk System provides a robust solution for library management by automating the processes of borrowing and returning books. The integration of book images and a user-friendly interface enhances user experience. Future iterations could include features like advanced search, user activity tracking, and integration with cloud databases for broader access.

11. Further Enhancements:

- Implementing a cloud-based database for remote access.
- Adding an advanced search and filtering mechanism for books.
- Providing analytics for user borrowing trends.

12. Bibliography:

- Tkinter Documentation: <https://docs.python.org/3/library/tkinter.html>
- JSON Documentation: <https://docs.python.org/3/library/json.html>
- Pillow Documentation: <https://pillow.readthedocs.io/en/stable/>
- Open Library Covers API: <https://openlibrary.org/dev/docs/api/covers>