National Academy of Science and Technology

(Affiliated to Pokhara University)

Accredited by University Grants Commission (UGC), Nepal (2022)

Uttarbehedi 4, Dhangadhi, Kailali, Nepal

A
Project Report
On
Old Book Store

For the partial fulfillment of the requirements for the degree of Bachelor of Computer Engineering Under Pokhara University

Submitted to

Department of Computer Engineering

National Academy of Science and Technology

Under the supervision of

Mr. Sunil Bahadur Bist

Lecturer, Department of Computer Engineering

Submitted by

Birendra Chaudhary (22070200)
Dilli Raj Bhatta (22070202)
Kaustubh Pant (22070217)
Santosh Rana (22070232)

BE Computer, 6th Semester
November, 2025

DECLARATION

We, **Birendra Chaudhary, Dilli Raj Bhatta, Kaustubh Pant, Santosh Rana**, students of **B.E. Computer, 6th semester**, NAST affiliated to Pokhara University, hereby declare that work undertaken in this minor project entitled **"OLD BOOK STORE"** is the result of our own work and research, carried out under the guidance of **Mr. Sunil Bahadur Bist**.

This project has not been submitted previously, either in part or in full, for the award of any degree, diploma or certificate at any other institution or university.

We further declare that all information and data used in this project have been obtained through genuine sources, and due acknowledgment has been given wherever applicable.

....	....
Birendra Chaudhary	Dilli Raj Bhatta
Date: 2025/11/13	Date: 2025/11/13

....	....
Kaustubh Pant	Santosh Rana
Date: 2025/11/13	Date: 2025/11/13

ACKNOWLEDGEMENT

Any successful academic project is the result of combined efforts, guidance, and encouragement from many individuals. We would like to take this opportunity to express our heartfelt gratitude to all who contributed to the successful completion of this project.

First and foremost, we sincerely thank **Mr. Sunil Bahadur Bist**, our project supervisor, Department of Computer Engineering, NAST, for his continuous guidance, support, and valuable suggestions throughout the development of this project. His mentorship helped us navigate challenges and provided insightful direction at every stage.

We extend our gratitude to the faculty of the **Department of Computer Engineering, NAST**, for providing the necessary resources, encouragement, and a conducive environment for learning and experimentation.

We are also thankful to our families and friends for their moral support, patience, and motivation during the project. Their encouragement helped us remain focused and committed throughout the course of the work.

Finally, we acknowledge all the authors, researchers, and online resources whose works inspired and guided us in designing and implementing the Old Book Store system.

We hope that the outcomes of this project will contribute meaningfully to sustainable book reuse and provide a foundation for future enhancements in online book trading systems.

Team member

Birendra Chaudhary (22070200)

Dilli Raj Bhatta (22070202)

Kaustubh Pant (22070217)

Santosh Rana (22070232)

ABSTRACT

The Old Book Store project is a web-based application developed to facilitate the online buying and selling of used books through a convenient and user-friendly platform. The system enables users to register, log in, and manage their accounts, allowing them to list their old books for sale or purchase books from other users. The primary goal of the project is to promote the reuse of books, reduce costs for readers, and encourage sustainable reading practices by minimizing paper waste.

The application provides essential features such as book search, wishlist management, order placement, and simulated payment processing. It also includes an integrated messaging system for communication between buyers and sellers. An administrative dashboard allows the admin to monitor user activities, manage books and transactions, and ensure the smooth operation of the system.

The project ensures a lightweight, scalable, and user-friendly web application. Overall, the Old Book Store project demonstrates how technology can enhance book circulation, support sustainable practices, and simplify online book trading.

Keywords: Used book, Oldbook Shop, Online Book Exchange, Bookstore Online

LIST OF ABBREVIATIONS

AI: Artificial Intelligence

CSS: Cascading Style Sheets

CSRF: Cross-Site Request Forgery

E-commerce: Electronic Commerce

E-Library: Electronic Library

ERD: Entity Relationship Diagram

HTML: Hyper Text Markup Language

IEEE: Institute of Electrical and Electronics Engineers

MVC: Model View Controller

OLX: Online eXchange

PHP: Hypertext Preprocessor

UI: User Interface

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Background

In recent years, the digital revolution has transformed how people buy and sell goods, including books. Traditional bookstores, though valuable, often face challenges such as limited physical space, inventory management issues, and restricted access for users in distant locations. To address these challenges, online book platforms have emerged as a convenient and efficient solution.

The Old Book Store project is developed to provide a web-based platform that facilitates the buying and selling of both new and used books. It aims to connect book lovers, readers, and sellers in a single, user-friendly environment that promotes accessibility, sustainability, and affordability in book trading

1.2 Objectives

The primary goals of this project are:

- To develop a user-friendly online platform for buying and selling old books.
- To implement secure user authentication.
- To enable users to manage wishlists, orders, and messages seamlessly.
- To create an admin dashboard for monitoring user activities, book listings, and transactions.

1.3 Purpose, Scope, and Applicability

1.3.1 Purpose

The purpose of this project are:

- To create a digital platform that simplifies the process of buying and selling old books.
- To promote the reuse and recycling of books, reducing waste and supporting sustainability.

- To provide an accessible, user-friendly environment where book enthusiasts can interact and exchange books conveniently.

1.3.2 Scope and Limitation

- The project covers the design and development of a web-based system for online book trading.
- Includes features such as user registration and authentication, book management, wishlist, orders, messaging, and simulated payment.
- Provides an admin dashboard for monitoring user activities, managing books, and updating order/payment statuses.
- Limited to simulated payments only no real payment gateway integration.
- Does not include real world delivery or logistics functionality.
- Intended for educational and demonstration purposes but can be extended for commercial use.

1.3.3 Applicability

- Useful for small-scale book exchange systems within colleges, communities, or local reading groups.
- Can serve as a model for developers building e-commerce or book-sharing platforms.
- Applicable as a project for academic learning, particularly in web development, database management, and software engineering.
- May be adapted for other product exchange platforms with minor modification.

1.4 Achievements

Through this project, a fully functional prototype of an online bookstore was successfully developed. The system enables secure login and registration, book uploads with details and images, and order and wishlist management. The admin panel provides real time oversight of all system activities, ensuring transparency and efficiency. The platform effectively demonstrates practical applications of web development, database management, and user interface design in solving real-world problems.

1.5 Organization of Report

The report is organized into seven chapters, each describing a specific stage of the Old Book Store project. Chapter 1 introduces the project by presenting its background, objectives, scope, and overall achievements. Chapter 2 provides a review of related literature and technologies that guided the development of the system. Chapter 3 discusses the system requirements, including both functional and non-functional aspects, along with project analysis and planning. Chapter 4 focuses on the system design, covering the architecture, database schema, and user interface design. Chapter 5 explains the implementation process and testing methodologies used to ensure system reliability and performance. Chapter 6 presents the results, testing outcomes, and user documentation, followed by Chapter 7, which concludes the report with key findings, significance, and recommendations for future improvements and system enhancements.

CHAPTER 2

SURVEY OF TECHNOLOGIES

This chapter provides a literature review of similar or relevant systems related to the concept of the Old Book Store project. The review includes academic papers, existing systems, and case studies that have addressed the reuse and online selling of second-hand books or related inventory management platforms. The analysis focuses on the technologies, architectures, and user functionalities used in these systems, aiming to identify gaps and derive best practices for the proposed project.

2.1 Review of Relevant Literature and Projects

Several online platforms and academic studies have explored the reuse and redistribution of educational resources, particularly second-hand books. These works demonstrate the potential of digital marketplaces in improving access to learning resources while promoting environmental sustainability. The analysis of these studies provides insights into the design, functionality, and technological considerations that can inform the development of the Old Book Store system.

2.1.1 ReBook: A Marketplace for Reused Books

A study by R. Bhatt and A. Patel, titled *"ReBook: A Sustainable Platform for Reusing Books Among Students"* [1], proposed a web application that allows students to exchange used books within a college network. This system was developed using PHP and MySQL and emphasized community engagement and environmental sustainability. The platform allowed users to list books, browse available titles, and connect with peers for exchange. While ReBook effectively fostered community participation, it was limited to a single college network and lacked scalability for larger user bases.

2.1.2 An Online Book Store System Based on Django Framework

Zhang et al. [2] proposed an online bookstore using the Django framework in Python. This system incorporated a robust admin panel, category-based book browsing, and order processing functionality. It demonstrated the effectiveness of Python-based frameworks in developing scalable and secure web applications. However, the system primarily focused on commercial book selling rather than promoting the reuse of educational resources, leaving room for adaptation in a sustainability-oriented context.

2.1.3 OLX and Quikr: Case Studies on General Second-hand Platforms

General online resale platforms, such as OLX and Quikr, support the buying and selling of various goods, including books. According to S. Desai and M. Shah [3], these platforms prioritize user-generated listings and informal, chat-based communication for completing transactions. While these systems are widely used, they are not specifically designed for book exchange and lack structured management of inventory, orders, and user roles. Their unstructured nature often limits efficiency and reliability in a dedicated book marketplace..

2.1.4 E-Library and Used Bookstore Model for Schools

An academic project presented by P. Mishra et al. [4] at the 2019 IEEE International Conference on Computing, Communication, and Security proposed a dual model combining an e-library and a second-hand book repository for schools. This project highlighted the potential of integrating digital libraries with physical resource management to improve access to educational materials. However, the system's scope was limited to schools, and it did not provide a scalable framework suitable for broader community use or commercial deployment.

Overall, the reviewed literature and case studies reveal that while multiple platforms support second-hand book transactions, few offer a focused, scalable, and structured experience tailored specifically for books. Based on these studies, the Old Book Store project is designed to fill this niche by incorporating user-friendly book listing and search features, a structured database model for managing books, orders, and users, lightweight web technologies such as Python Flask, MYSQL, HTML, and CSS, as well as role-based access and secure data handling.

CHAPTER 3

REQUIREMENTS AND ANALYSIS

3.1 Problem Definition

In traditional book-selling and exchange systems, users face multiple challenges such as limited access to second-hand books, inefficient inventory management, and lack of a structured platform for buying and selling used books. Physical bookstores often have space constraints, and community-based exchanges are typically informal, making it difficult to track orders or manage transactions. Additionally, existing online marketplaces either focus on a wide variety of products or are designed for commercial purposes, which may not address the needs of students, book enthusiasts, or small communities. The Old Book Store project addresses these problems by creating a dedicated online platform that facilitates the structured buying and selling of old books, streamlines order processing, manages user data securely, and promotes sustainability in book reuse.

3.2 Requirement Specification

3.2.1 Functional Requirements

- Users can register and log in securely.
- Users can list books for sale with details including title, author, price, and images.
- Users can search for books, add them to a wishlist, and place orders.
- An admin panel is available to monitor users, manage book listings, and update order or payment status.
- Users can send and receive messages regarding book transactions.

3.2.2 Non-Functional Requirements

- The system should be reliable and available online with minimal downtime.
- Data security must be ensured through secure login and password encryption.
- The user interface should be intuitive, responsive, and accessible across devices.
- The database must maintain data integrity and support efficient retrieval of information.

3.3 Planning and Scheduling

The development of the “Old Book Store” is guided by the Software Development Life Cycle (SDLC), and the schedule is divided into distinct stages with estimated timelines.

Start Date: 2025 May 8

End Date: Mid November 2025

ACTIVITIES	May			June				July				August-October				November	
	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2
Project Analysis	■																
Proposal Submission	■																
Proposal Defense		■															
Feasibility Study		■	■														
Designing			■														
Frontend Coding				■	■	■											
Mid-Term Defense							■										
Backend Coding								■	■	■	■	■	■	■			
Testing												■	■	■	■	■	
Final Testing																■	■
Documentation	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Final Defense																	■

Table 3.1 Gantt Chart

3.4 System Requirements

The system must meet specific hardware and software requirements for both development and deployment.

3.4.1 Hardware Requirements

- Processor: Intel Core i3 or higher
- RAM: Minimum 4 GB
- Storage: At least 10 GB of free disk space
- Internet Connection: Required for online testing and email verification

3.4.2 Software Requirements

- Operating System: Windows 10 or Linux
- Web Browser: Google Chrome, Mozilla Firefox
- Backend Framework: Python Flask
- Database: MySQL
- Development Environment: Visual Studio Code or PyCharm
- Libraries and Tools: Flask-Mail, Jinja2, Werkzeug

3.5 Preliminary Product Description

The Old Book Store system is a web-based application that allows users to buy and sell second-hand books efficiently. It offers functionalities such as secure user registration and login, book listing with detailed information, search and filter options, order placement, wishlist management, and messaging between buyers and sellers. The admin has control over the entire system through a dashboard that monitors user activities, manages book inventory, and updates payment or order statuses. The system is developed using lightweight technologies including Python Flask for the backend, HTML and CSS for frontend design, and MySQL for database management. This preliminary product description highlights the system's focus on usability, scalability, and sustainability, ensuring that it meets the needs of students, book enthusiasts, and small community users.

CHAPTER 4

DESIGN

4.1 Introduction

The design phase is a critical step in software development, serving as the blueprint for building the system. It translates the gathered requirements into structured components such as system architecture, database schema, and user interfaces. In the this project, the design aims to ensure modularity, scalability, and usability while addressing all functional requirements.

This chapter elaborates on three key components:

1. System Design
2. Database Design
3. Interface (UI/UX) Design

4.2 System Design

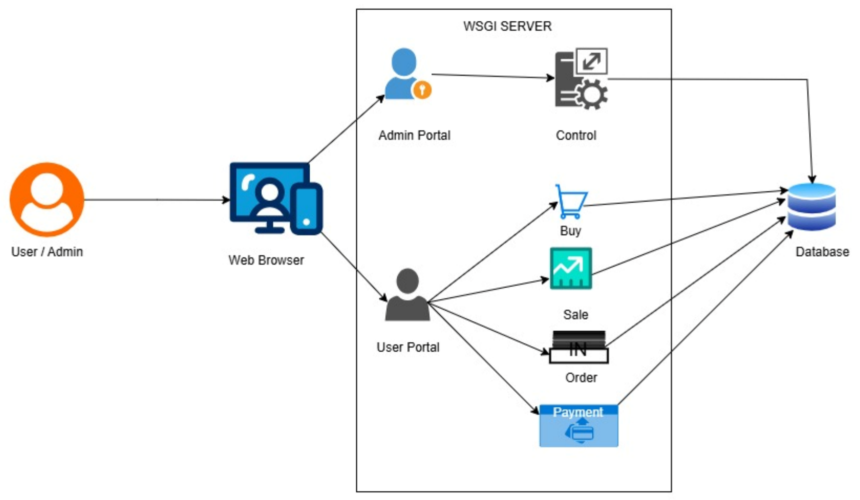


Figure 4.1 System Architecture

4.3 Database Design

ER Diagram:

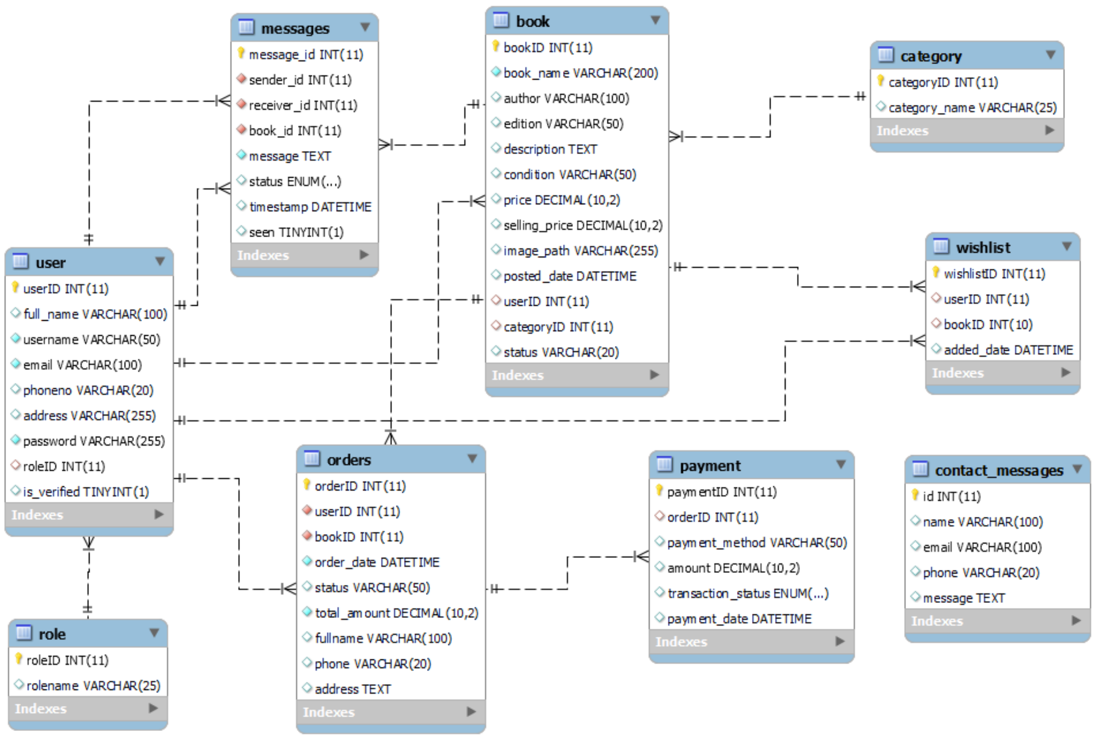


Figure 4.2: ER Diagram

4.4 Interface Design

Interface design focuses on creating user-friendly and visually consistent screens for both users and administrators.

User Interface:

- Home Page: Displays categories and featured books with a navigation bar for easy access.
- Registration/Login Page: Allows users to create an account or log in securely.
- Buy Books Page: Users can browse, search, and purchase available books.
- Sell Books Page: Provides a form to upload book details and images.
- Wishlist Page: Displays saved books for future purchase.
- Orders Page: Shows user orders with order and payment status.
- Messages Page: Enables users to chat with buyers or sellers directly.

Admin Interface:

- Admin Dashboard: Displays system statistics (total users, books, orders).
- User Management Page: Allows viewing or deleting users.
- Book Management Page: Displays all uploaded books and lets admin remove any inappropriate listings.
- Payment Management Page: Allows updating payment status for any order.

4.5 Summary

This chapter outlined the complete system design of the Old Book Store application, covering its architectural structure, database schema, and user interface. The system was designed to ensure modularity, scalability, and usability, enabling both buyers and sellers to interact seamlessly. The database model supports relational integrity, while the user interfaces offer simplicity and efficiency. These design elements collectively form the foundation for the system's successful implementation and testing described in subsequent chapters

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Implementation Approaches

The implementation of the Old Book Store project followed a modular development approach. The system was divided into different modules such as User Management, Book Management, Order and Payment System, and Admin Panel. Each module was developed and tested independently before integration. The backend was implemented using Python Flask, and the frontend used HTML, CSS, and JavaScript for dynamic and responsive design. The MySQL database was used to store and retrieve all book, user, and order-related information efficiently.

5.2 Coding Details and Code Efficiency

The code is structured using the Model-View-Controller (MVC) design pattern to separate logic, data, and user interface. Flask routes handle user requests, models interact with the database, and templates render dynamic content to users.

5.2.1 Code Efficiency

- Reusable functions were used for login, registration, and database operations.
- Passwords are stored using hashing algorithms (Werkzeug) for security.
- Database queries are optimized with indexed fields and minimal joins.
- Static resources (CSS, JS, images) are cached to reduce page loading time.
- Error handling and validation reduce server crashes and improve stability**.**

5.3 Testing Approach

The system underwent multiple stages of testing to ensure reliability, functionality, and performance. Testing was done module by module and then integrated to verify interaction between components.

5.3.1 Unit Testing

Individual modules such as user registration, login, and book upload were tested independently. Unit testing verified that each function performed as intended without affecting other parts of the system.

5.3.2 Integrated Testing

After unit testing, all modules were integrated and tested together. This confirmed that data flowed correctly between modules such as order processing, payment, and admin monitoring.

5.3.3 Beta Testing

The system was deployed for testing among a small group of users who interacted with it under real-world conditions. Feedback was collected regarding usability, interface design, and performance, and necessary improvements were implemented.

5.4 Modifications and Improvements

- Improved the admin dashboard to manage payment and order status dynamically.
- Added email verification and password reset functionality for user accounts.
- Enhanced responsive design for better mobile and tablet usability.
- Optimized database performance for faster query execution.
- Implemented input validation and error messages for user actions.

5.5 Test Cases

Test Case ID	**Test Scenario**	**Input**	**Expected Output**	**Result**
TC_01	Register with valid details	Valid name, username, email, password, phone, address	Account created successfully and verification email sent	Pass
TC_02	Register with existing email	Existing email address	Error message: “Email already registered”	Pass
TC_03	Login with correct credentials	Valid email and password	Redirect to user dashboard	Pass

TC_04	Login with incorrect password	Wrong password	Error message: "Invalid email or password"	Pass
TC_05	Forgot password (valid email)	Registered email	Password reset link sent to email	Pass
TC_06	Forgot password (invalid email)	Unregistered email	Error message: "Email not found"	Pass
TC_07	Upload book (Users)	Valid book details and image file	Book uploaded successfully	Pass
TC_08	Add book to Wishlist	Click "Add to Wishlist"	Book added to Wishlist	Pass
TC_09	Remove book from Wishlist	Click "Remove"	Book removed from Wishlist successfully	Pass
TC_10	Place order (valid details)	Valid payment info	Order confirmed and status = "Pending Payment"	Pass
TC_11	Payment completed (admin)	Update payment status	Status changes to "Completed"	Pass
TC_12	Access admin panel as normal user	Logged-in user role = user	Access denied page shown	Pass
TC_13	Access admin panel as admin	Logged-in admin	Admin dashboard displayed	Pass

TC_14	Search book	Search keyword like “Python”	Matching books displayed	Pass
TC_15	Logout	Click logout	Session cleared, redirected to login	Pass

Table 5.1 Test Case

CHAPTER 6

RESULTS AND DISCUSSION

This chapter presents the results obtained after the successful implementation and testing of the Old Book Store system. It includes an overview of the system's performance, the outcomes of functional testing, and user documentation describing how the system can be used effectively.

6.1 Test Reports

After completing unit, integration, and beta testing, the system was found to perform efficiently across all modules. The main modules tested were User Authentication, Book Management, Wishlist, Order and Payment Simulation, Messaging, and Admin Management. All functional test cases produced the expected results as outlined in Chapter 5. Below is a summary of the testing outcomes:

- All user and admin test cases passed successfully.
- The system demonstrated stable performance under normal load conditions.
- The login and registration modules operated with secure validation and proper error handling.
- The book upload and order modules performed all CRUD (Create, Read, Update, Delete) operations accurately.
- Data integrity was maintained across all database tables during book selling, order cancellation, and payment updates.
- The admin panel successfully provided full control over users, books, and payment status updates.

6.2 User Documentation

This section provides guidance for users on how to interact with the system efficiently.

For Normal Users:

1. **Registration and Login**
    - o New users must register with their name, email, and password.
    - o After registration, a verification email is sent. Users must verify their account before logging in.
2. **Buying Books**
    - o After logging in, users can browse available books in the Buy Books section.

- Search books by title.
- Click “Checkout & Pay” to place an order and complete simulated payment.

3. **Selling Books**
    - Navigate to the Sell Books section.
    - Fill in details such as title, author, category, price, and upload a book image.
    - Click Submit to list the book for sale.
4. **Wishlist Management**
    - Users can add books to their wishlist for later purchase.
    - Books can be removed from the wishlist.
5. **Order Management**
    - Users can view and cancel their orders from the Orders page.
    - Payment status and order history are displayed clearly.
6. **Messaging Feature**
    - Users can communicate directly with sellers/buyers via the Messages section.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATION

7.1 Conclusion

The Old Book Store project is a web-based platform designed to make buying and selling used books easier and more sustainable. It was built using the Python Flask framework with a MySQL database, and the front end was developed with HTML, CSS, and JavaScript to keep the interface clean and user-friendly.

The system allows users to register, upload books for sale, create wishlists, and manage their orders. It also includes an admin dashboard to oversee users, listings, and transactions, helping keep everything organized and running smoothly.

Beyond its technical side, the project aims to promote sustainability by encouraging people to reuse books instead of throwing them away. It also supports affordable access to learning materials, which can help students and readers in general.

7.2 Significance of the System

1. Environmental Significance
    - Encourages reuse and recycling of books, thereby reducing paper waste and supporting sustainable practices.
2. Economic Significance
    - Provides an affordable way for students and readers to access educational materials, reducing overall costs.
3. Educational Significance
    - Serves as a real-world example of applying web development and database concepts to solve meaningful problems.
4. Technical Significance
    - Demonstrates efficient use of Python Flask, relational databases, and MVC architecture in a scalable e-commerce application.

7.3 Recommendations

Based on the analysis and implementation experience, the following recommendations are proposed for future enhancement of the Old Book Store system:

1. Integration of Real Payment Gateways

- Implement secure payment options such as eSewa, Khalti, or Stripe for real-time transactions.

2. Logistics and Delivery Module
    - Incorporate a delivery tracking feature with integration of third-party courier APIs.
3. Mobile Application Development
    - Develop Android and iOS versions using Flutter or React Native to expand accessibility.
4. Improved Messaging and Notification System
    - Add real-time chat and push notifications to improve user interaction and response time.
5. Enhanced Security and Data Protection
    - Implement two-factor authentication (2FA) and advanced encryption for user data and transactions.

REFERENCES

[1] R. Bhatt and A. Patel, “ReBook: A Sustainable Platform for Reusing Books Among Students,” International Journal of Advanced Computer Science and Applications (IJACSA), vol. 12, no. 8, pp. 56–63, 2021.

[2] Y. Zhang, L. Wang, and H. Liu, “An Online Book Store System Based on Django Framework,” Proceedings of the 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), pp. 1193–1198, 2020.

[3] S. Desai and M. Shah, “Case Study: Online Resale Marketplaces like OLX and Quikr,” IEEE International Conference on E-Business Engineering, pp. 88–92, 2019.

[4] P. Mishra, A. Sinha, and R. Das, “E-Library and Used Bookstore Model for Schools,” Proceedings of the 2019 IEEE International Conference on Computing, Communication, and Security (ICCCS), pp. 512–517, 2019.

[5] Flask Documentation, “Flask Web Framework Documentation,” [Online]. Available: https://flask.palletsprojects.com/. [Accessed: Aug. 5, 2025].

[6] GitHub, “Flask-Based Bookstore Management System Repositories,” [Online]. Available: https://github.com/. [Accessed: Aug. 5, 2025].

[7] Stack Overflow, “Flask, SQL, and Web Development Discussions,” [Online]. Available: https://stackoverflow.com/ .[Accessed: Aug. 15, 2025].

ANNEX I

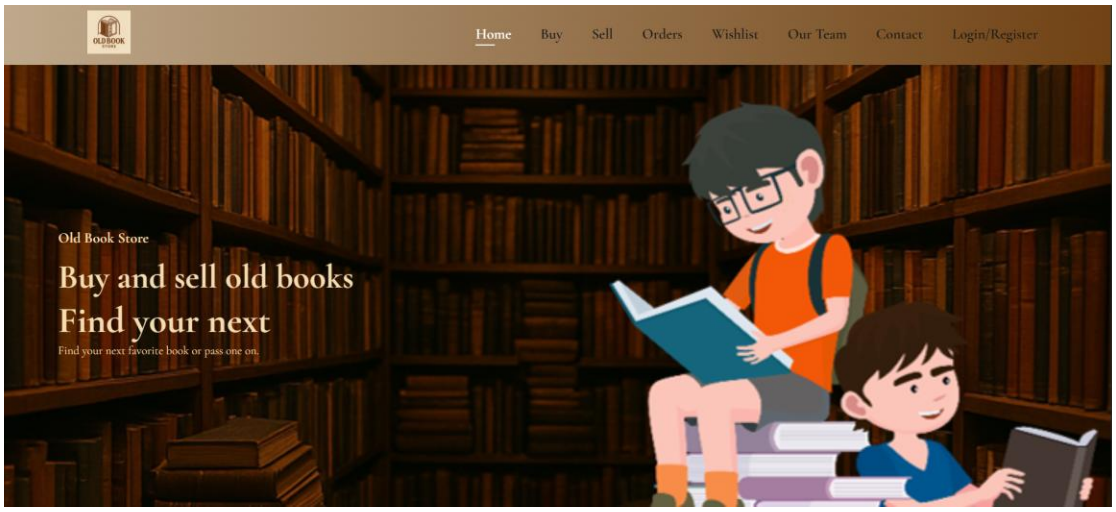


Figure Annex 1.1: Home page

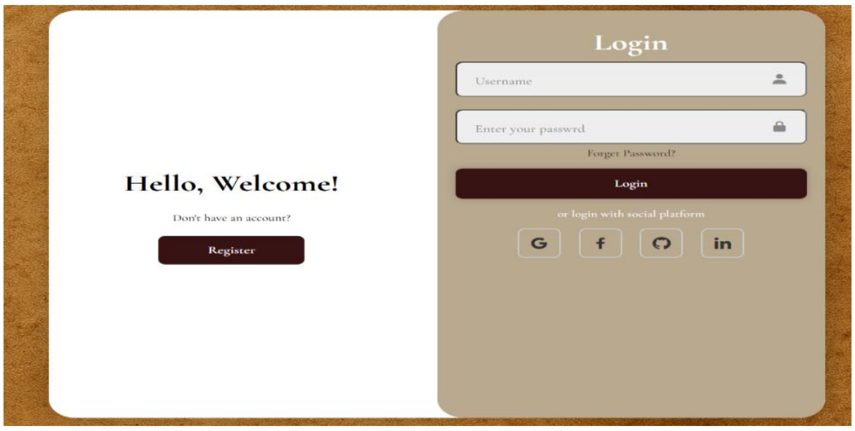


Figure Annex 1.2: Login Page

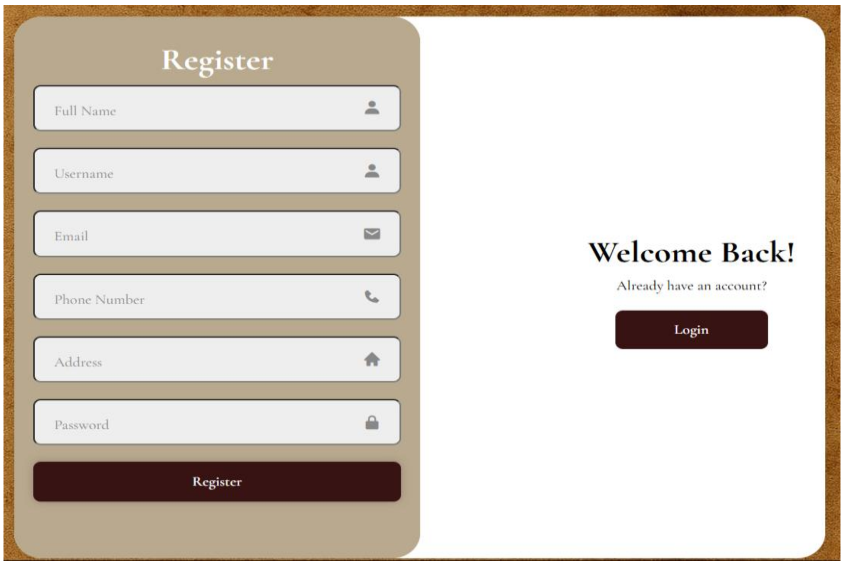
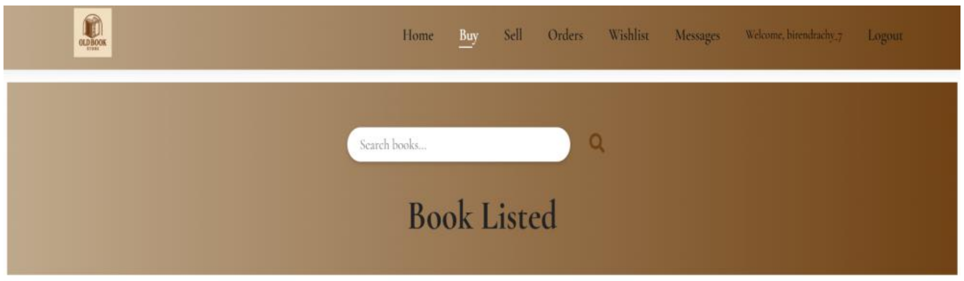
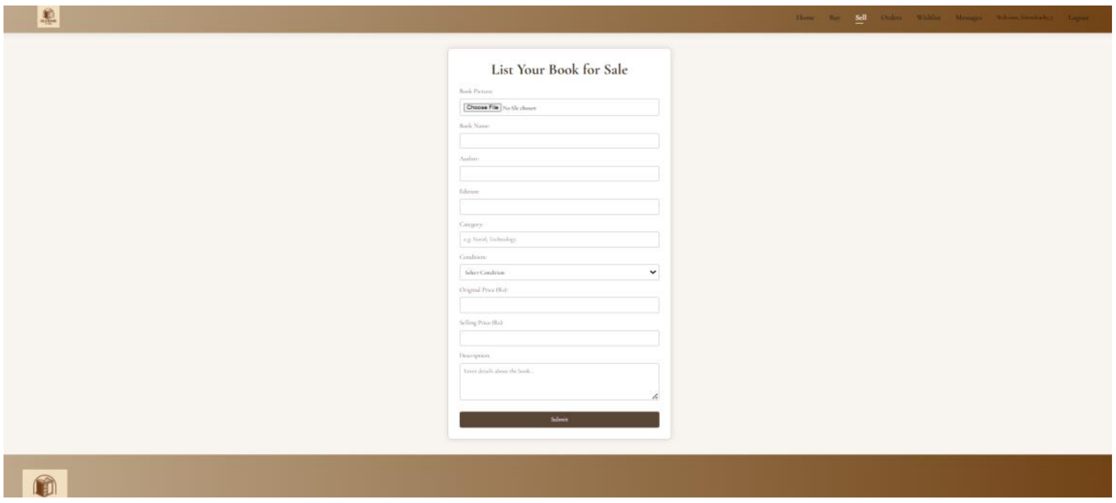


Figure Annex 1.3: Register Page



Book Listed

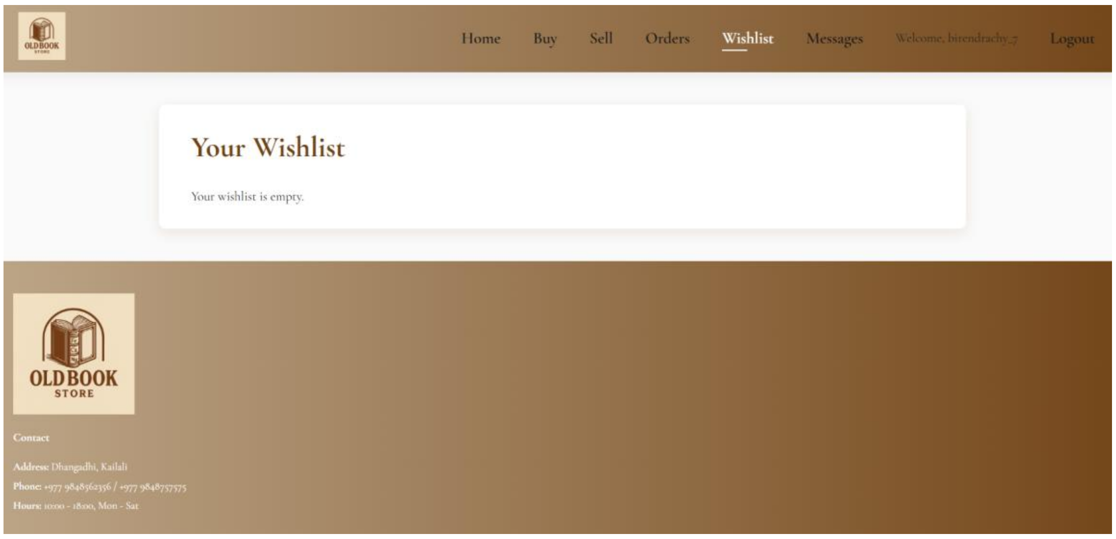
Figure Annex 1.4: Book listed Page



List Your Book for Sale

Book Picture:
Choose File No file chosen
Book Name:
Author:
Edition:
Category:
e.g. Novel, Technology
Condition:
Select Condition
Original Price (Rs):
Selling Price (Rs):
Description:
Enter details about the book...

Figure Annex 1.5: Sell Page



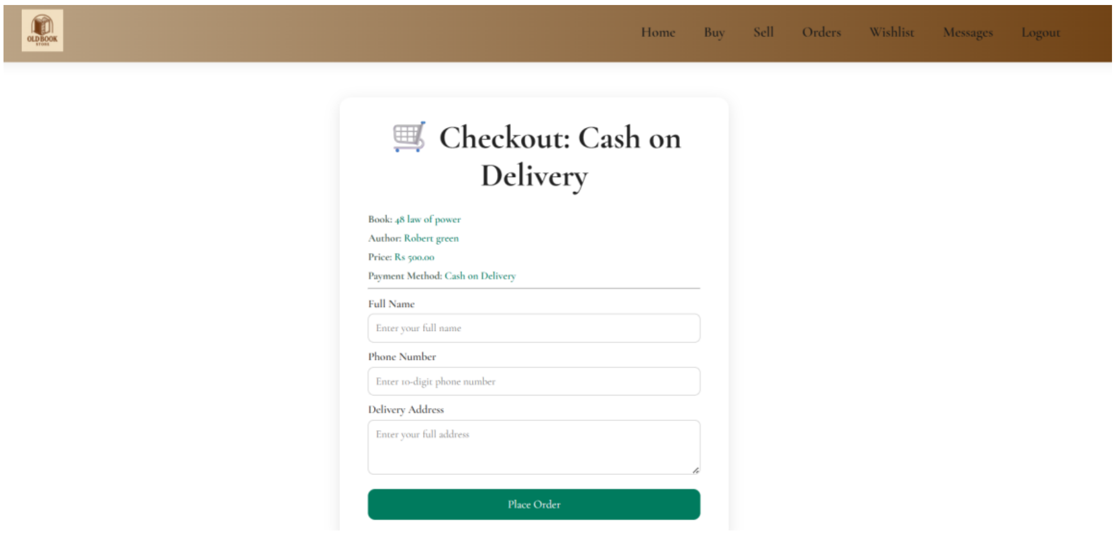
Your Wishlist

Your wishlist is empty.

Contact

Address: Dhangadhi, Kailali
Phone: +977 9848562356 / +977 9848757575
Hours: 10:00 - 18:00, Mon - Sat

Figure Annex 1.6: Wishlist Page

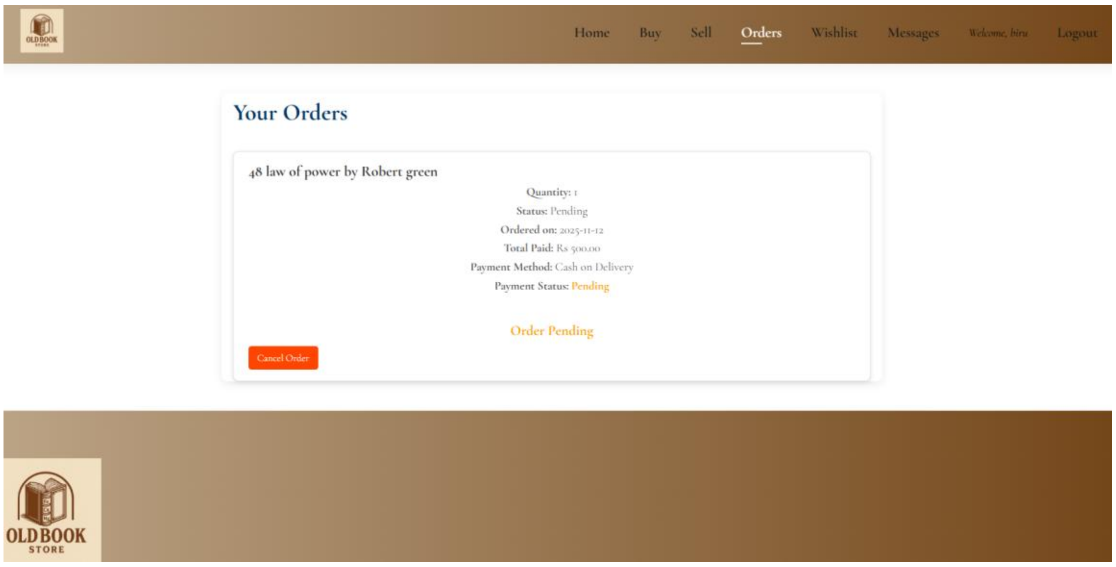


Checkout: Cash on Delivery

Book: 48 law of power
Author: Robert green
Price: Rs 500.00
Payment Method: Cash on Delivery

Full Name: Enter your full name
Phone Number: Enter 10-digit phone number
Delivery Address: Enter your full address

Figure Annex 1.7: Checkout Page

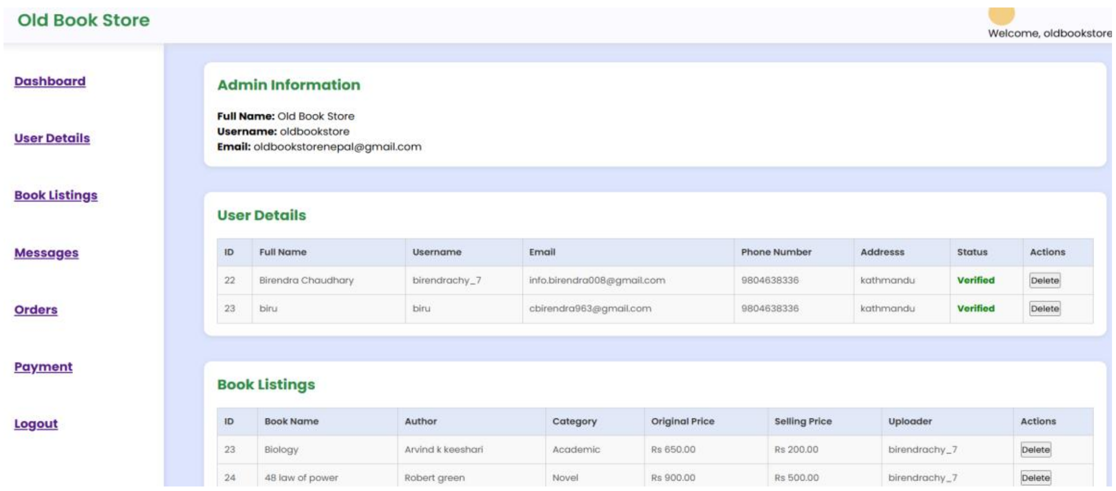


Your Orders

48 law of power by Robert green
Quantity: 1
Status: Pending
Ordered on: 2025-11-12
Total Paid: Rs 500.00
Payment Method: Cash on Delivery
Payment Status: Pending

Order Pending

Figure Annex 1.8: Order Page



Admin Information

Full Name: Old Book Store
Username: oldbookstore
Email: oldbookstorenepal@gmail.com

User Details

ID	Full Name	Username	Email	Phone Number	Addresss	Status	Actions
22	Birendra Chaudhary	birendrachy_7	info.birendra008@gmail.com	9804638336	kathmandu	Verified	Delete
23	biru	biru	cbirendra963@gmail.com	9804638336	kathmandu	Verified	Delete

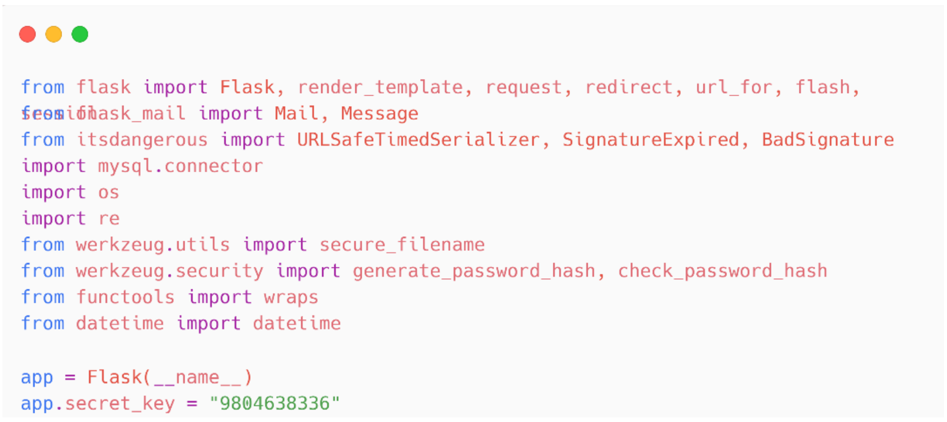
Book Listings

ID	Book Name	Author	Category	Original Price	Selling Price	Uploader	Actions
23	Biology	Arvind k keeshari	Academic	Rs 650.00	Rs 200.00	birendrachy_7	Delete
24	48 law of power	Robert green	Novel	Rs 900.00	Rs 500.00	birendrachy_7	Delete

Figure Annex 1.9: Admin Page

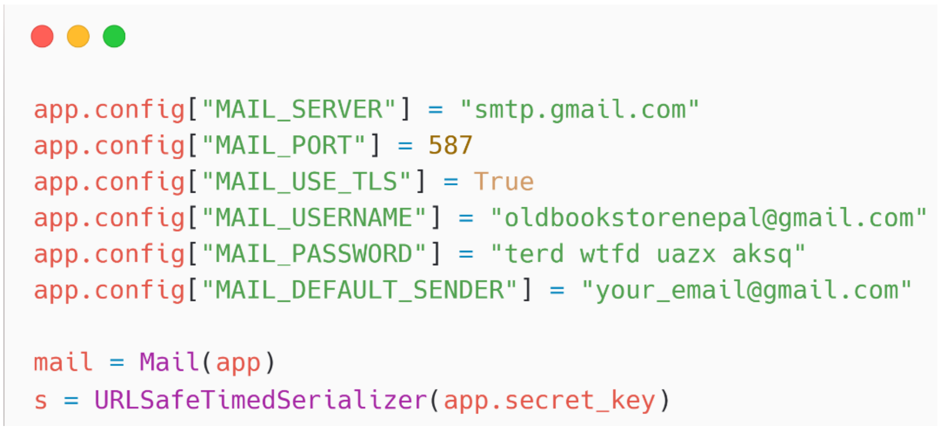
ANNEX II

Import and Configuration



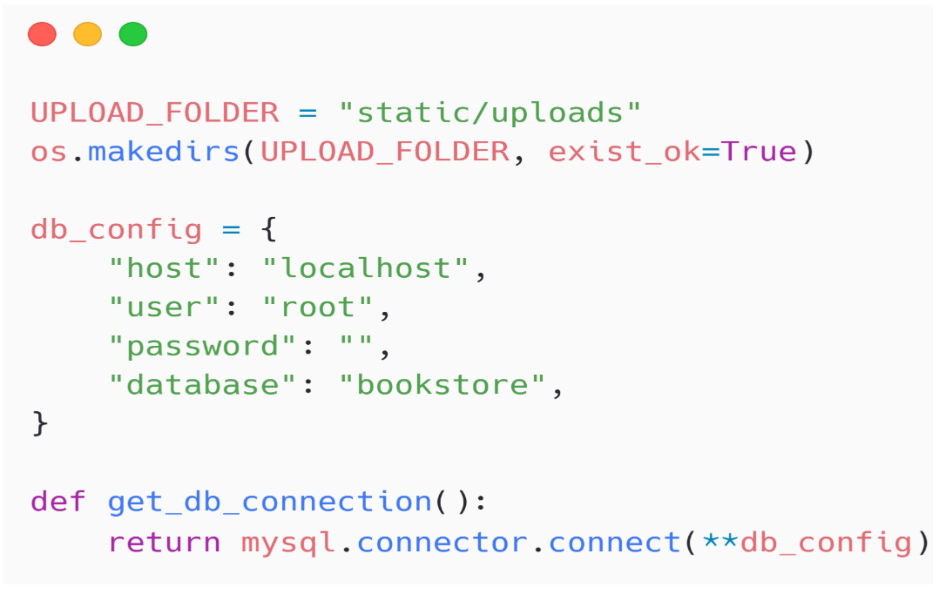
```
from flask import Flask, render_template, request, redirect, url_for, flash,
from flask_mail import Mail, Message
from itsdangerous import URLSafeTimedSerializer, SignatureExpired, BadSignature
import mysql.connector
import os
import re
from werkzeug.utils import secure_filename
from werkzeug.security import generate_password_hash, check_password_hash
from functools import wraps
from datetime import datetime

app = Flask(__name__)
app.secret_key = "9804638336"
```

Email Configuartion:



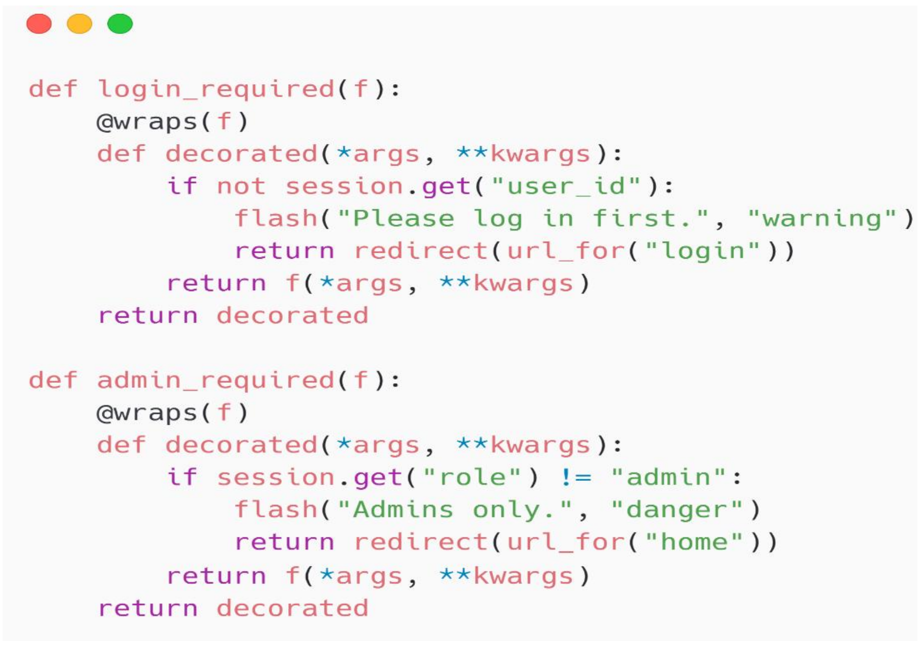
```
app.config["MAIL_SERVER"] = "smtp.gmail.com"
app.config["MAIL_PORT"] = 587
app.config["MAIL_USE_TLS"] = True
app.config["MAIL_USERNAME"] = "oldbookstorenepal@gmail.com"
app.config["MAIL_PASSWORD"] = "terd wtfd uazx aksq"
app.config["MAIL_DEFAULT_SENDER"] = "your_email@gmail.com"

mail = Mail(app)
s = URLSafeTimedSerializer(app.secret_key)
```

File Upload And Database Configuration:



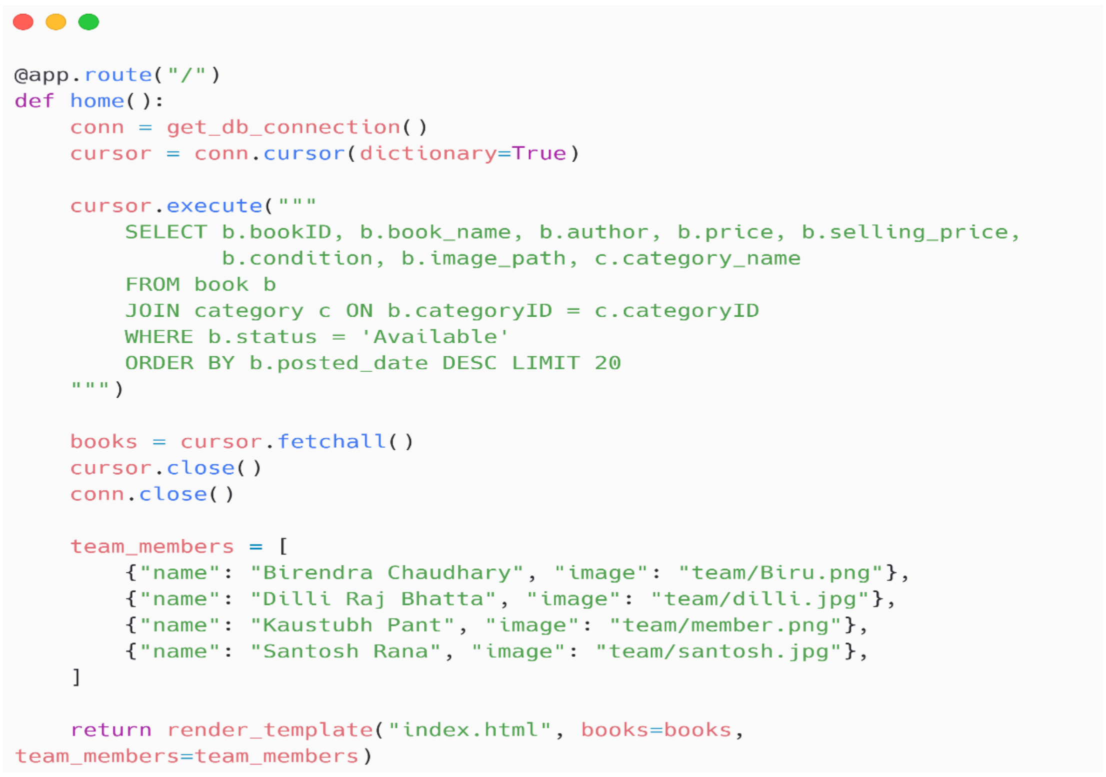
```
UPLOAD_FOLDER = "static/uploads"
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

db_config = {
     "host": "localhost",
     "user": "root",
     "password": "",
     "database": "bookstore",
}

def get_db_connection():
     return mysql.connector.connect(**db_config)
```

Authentication Decorators



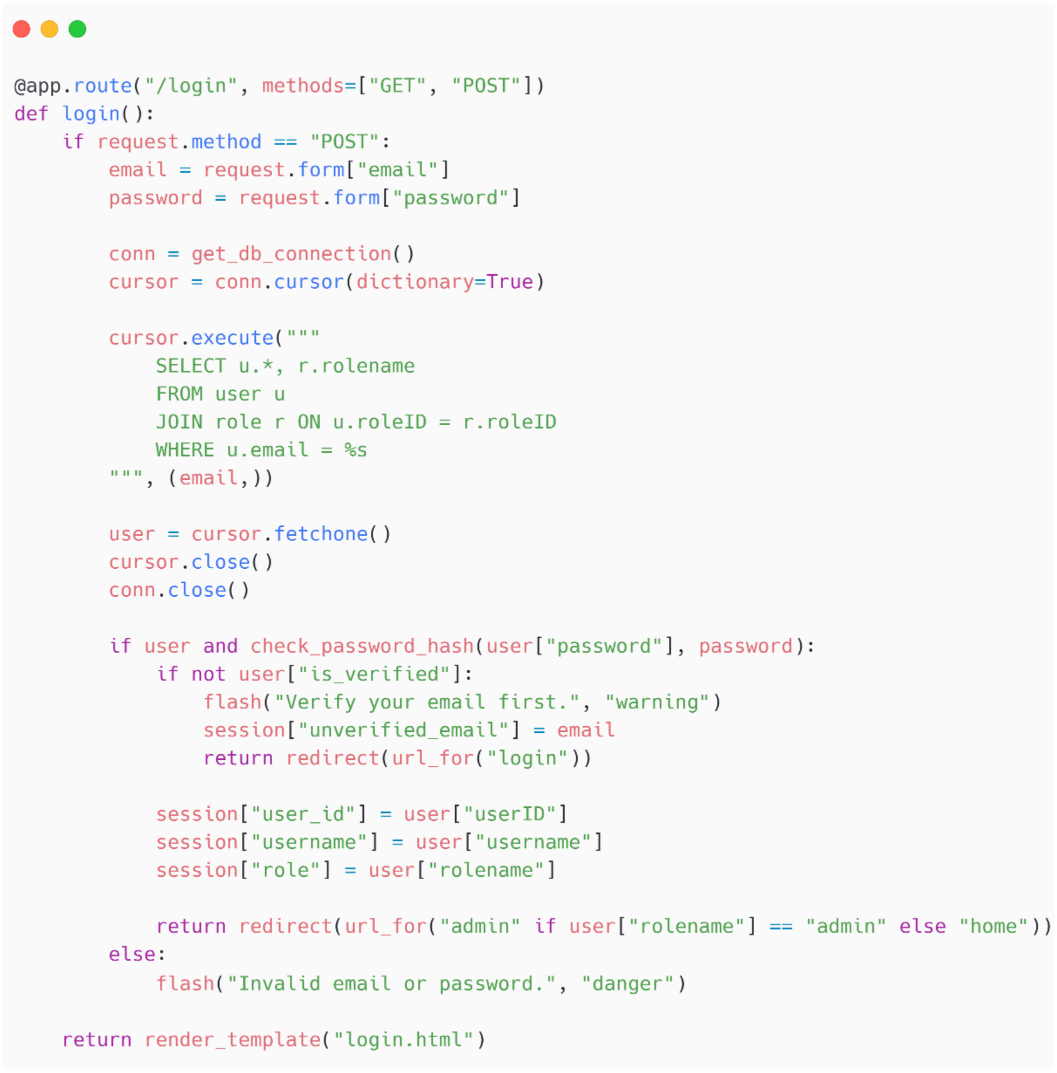
```python
def login_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        if not session.get("user_id"):
            flash("Please log in first.", "warning")
            return redirect(url_for("login"))
        return f(*args, **kwargs)
    return decorated

def admin_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        if session.get("role") != "admin":
            flash("Admins only.", "danger")
            return redirect(url_for("home"))
        return f(*args, **kwargs)
    return decorated
```

Home Route:



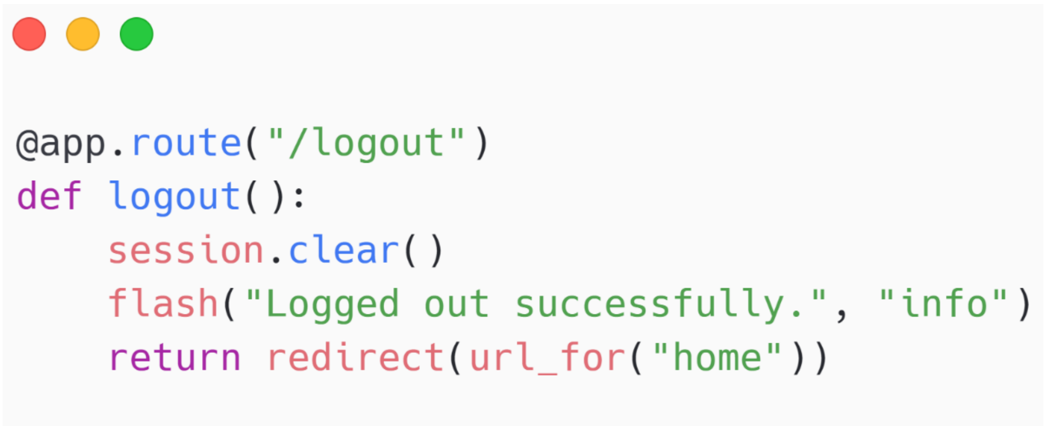
```python
@app.route("/")
def home():
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    cursor.execute("""
        SELECT b.bookID, b.book_name, b.author, b.price, b.selling_price,
               b.condition, b.image_path, c.category_name
        FROM book b
        JOIN category c ON b.categoryID = c.categoryID
        WHERE b.status = 'Available'
        ORDER BY b.posted_date DESC LIMIT 20
    """)

    books = cursor.fetchall()
    cursor.close()
    conn.close()

    team_members = [
        {"name": "Birendra Chaudhary", "image": "team/Biru.png"},
        {"name": "Dilli Raj Bhatta", "image": "team/dilli.jpg"},
        {"name": "Kaustubh Pant", "image": "team/member.png"},
        {"name": "Santosh Rana", "image": "team/santosh.jpg"},
    ]

    return render_template("index.html", books=books,
team_members=team_members)
```

Login Route:



```python
@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        email = request.form["email"]
        password = request.form["password"]

        conn = get_db_connection()
        cursor = conn.cursor(dictionary=True)

        cursor.execute("""
            SELECT u.*, r.rolename
            FROM user u
            JOIN role r ON u.roleID = r.roleID
            WHERE u.email = %s
        """, (email,))

        user = cursor.fetchone()
        cursor.close()
        conn.close()

        if user and check_password_hash(user["password"], password):
            if not user["is_verified"]:
                flash("Verify your email first.", "warning")
                session["unverified_email"] = email
                return redirect(url_for("login"))

            session["user_id"] = user["userID"]
            session["username"] = user["username"]
            session["role"] = user["rolename"]

            return redirect(url_for("admin" if user["rolename"] == "admin" else "home"))
        else:
            flash("Invalid email or password.", "danger")

    return render_template("login.html")
```

Logout Route:



```python
@app.route("/logout")
def logout():
    session.clear()
    flash("Logged out successfully.", "info")
    return redirect(url_for("home"))
```

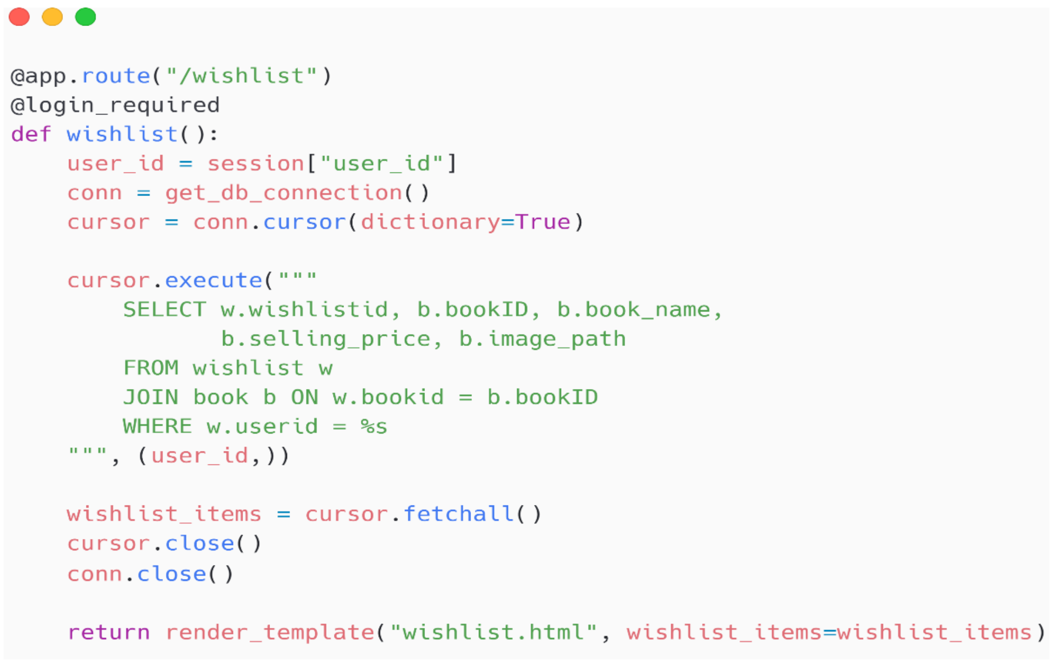
Email Verification:

```python
@app.route("/confirm_email/<token>")
def confirm_email(token):
    try:
        email = s.loads(token, salt="email-confirm-salt", max_age=3600)
    except SignatureExpired:
        flash("Link expired.", "danger")
        return redirect(url_for("register"))
    except BadSignature:
        flash("Invalid link.", "danger")
        return redirect(url_for("register"))

    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("UPDATE user SET is_verified = 1 WHERE email = %s",
(email,))
    conn.commit()
    cursor.close()
    conn.close()

    flash("Email verified successfully!", "success")
    return redirect(url_for("login"))
```

Sell Rotue:

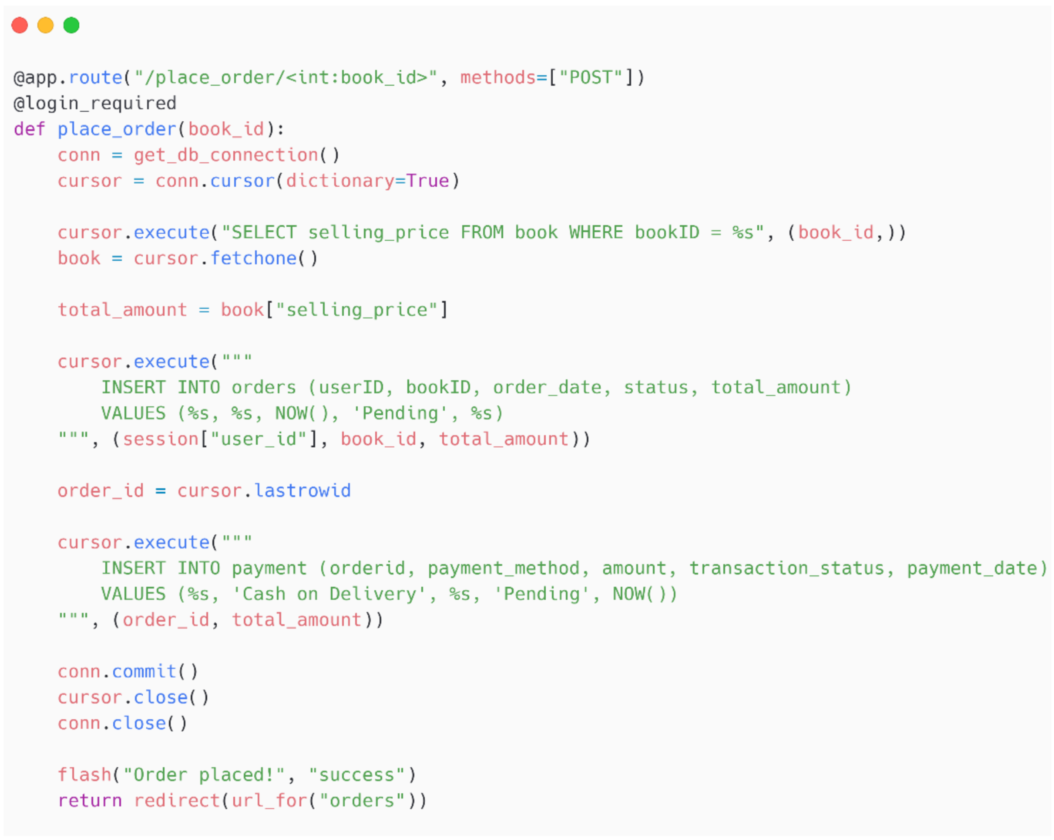
```python
@app.route("/sell", methods=["GET", "POST"])
@login_required
def sell():
    if request.method == "POST":
        book_name = request.form["book_name"].strip()
        author = request.form["author"].strip()
        edition = request.form.get("edition", "").strip()
        description = request.form.get("description", "").strip()
        condition = request.form["condition"]
        category_name = request.form["category"].strip()

        original_price = float(request.form["original_price"])
        selling_price = float(request.form["selling_price"])

        image = request.files.get("book_image")
        filename = secure_filename(image.filename)
        image_path = os.path.join(UPLOAD_FOLDER, filename)
        image.save(image_path)
        relative_path = f"uploads/{filename}"

        conn = get_db_connection()
        cursor = conn.cursor()

        cursor.execute("SELECT categoryID FROM category WHERE category_name = %s",
(category_name,))
        category = cursor.fetchone()

        category_id = category[0] if category else cursor.lastrowid

        cursor.execute("""
            INSERT INTO book (book_name, author, edition, description,
                condition, price, selling_price, image_path, posted_date,
                userID, categoryID, status)
            VALUES (%s, %s, %s, %s, %s, %s, %s, %s, NOW(), %s, %s, 'Available')
        """, (book_name, author, edition, description, condition, original_price,
              selling_price, relative_path, session["user_id"], category_id))

        conn.commit()
        cursor.close()
        conn.close()

        flash("Book listed successfully!", "success")
        return redirect(url_for("sell"))

    return render_template("sellbook.html")
```

Wishlist Route:



```python
@app.route("/wishlist")
@login_required
def wishlist():
    user_id = session["user_id"]
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    cursor.execute("""
        SELECT w.wishlistid, b.bookID, b.book_name,
               b.selling_price, b.image_path
        FROM wishlist w
        JOIN book b ON w.bookid = b.bookID
        WHERE w.userid = %s
    """, (user_id,))

    wishlist_items = cursor.fetchall()
    cursor.close()
    conn.close()

    return render_template("wishlist.html", wishlist_items=wishlist_items)
```

Place Order Route:



```python
@app.route("/place_order/<int:book_id>", methods=["POST"])
@login_required
def place_order(book_id):
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    cursor.execute("SELECT selling_price FROM book WHERE bookID = %s", (book_id,))
    book = cursor.fetchone()

    total_amount = book["selling_price"]

    cursor.execute("""
        INSERT INTO orders (userID, bookID, order_date, status, total_amount)
        VALUES (%s, %s, NOW(), 'Pending', %s)
    """, (session["user_id"], book_id, total_amount))

    order_id = cursor.lastrowid

    cursor.execute("""
        INSERT INTO payment (orderid, payment_method, amount, transaction_status, payment_date)
        VALUES (%s, 'Cash on Delivery', %s, 'Pending', NOW())
    """, (order_id, total_amount))

    conn.commit()
    cursor.close()
    conn.close()

    flash("Order placed!", "success")
    return redirect(url_for("orders"))
```