K.RAMAKRISHNAN
COLLEGE OF TECHNOLOGY
An Autonomous Institution
Affiliated to Anna University Chennai, Approved by AICTE New Delhi,
ISO 9001:2015 & ISO 14001:2015 Certified Institution, Accredited with 'A+' grade by NAAC
Samayapuram, Tiruchirappalli – 621 112, Tamilnadu, India.

A Project Report

on

ONLINE BOOK STORE

Submitted in partial fulfillment of requirements for the award of the course

of

EGB1201 – JAVA PROGRAMMING

Under the guidance of

Ms. Hema R., M.E.,

Assistant Professor / Information Technology

Submitted By

PRIYADHARSHINI C (ECB23080)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(Autonomous)

TRICHY - 621112

DECEMBER 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(Autonomous Institution affiliated to Anna University, Chennai)

TRICHY - 621112

BONAFIDE CERTIFICATE

Certified that this project report on **"ONLINE BOOK STORE"** is the Bonafide work of **PRIYADHARSHINI C (ECB23080)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

Signature

Ms. HEMA R., M.E.,

SUPERVISOR,

Department of Information Technology,

K. Ramakrishnan College of Technology,

Trichy – 621112

Signature

Dr. SYEDAKBAR S., M.E., Ph.D.,

HEAD OF THE DEPARTMENT,

Department of ECE,

K. Ramakrishnan College Technology,

Trichy – 621112

Submitted for the viva-voice examination held on 07.12.24

INTERNAL EXAMINER

EXTERNAL EXAMINER

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education

MISSION OF THE INSTITUTION

- Produce smart technocrats with empirical knowledge who can surmount the global challenges
- Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
- Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

VISION OF THE DEPARTMENT

To create innovative and socially responsible Electronics and Communication Engineers with design skills and research focus to meet Societal and Industrial needs.

MISSION OF THE DEPARTMENT

- M1: To provide high quality education and professional ethics to students through enhanced learning environment
- M2: To impart a creative environment towards centre of excellence in department with design skill and exposure for research.
- M3: To nurture required employable skills of students to satisfy the industry and social needs with ethical and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

- PEO1: Core Knowledge Development: Graduates will have enhanced engineering skills in the field of electronics, communication and interdisciplinary areas to serve the society with global standards.

- PEO2: Professional development: Graduates will apply the technical knowledge for continuous up gradation of their professional skills to become an inimitable employee, researcher or entrepreneur.
- PEO3: Analytical Thinking: Graduates will have analytic and thinking skills to provide the innovative solutions for industry and societal requirements.

PROGRAM OUTCOMES

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- PSO1: To analyse, design and develop solutions by applying foundational concepts of electronics and communication engineering.
- PSO2: To apply design principles and best practices for developing quality products for scientific and business applications.

ABSTRACT

The Online Bookstore Project is a Java-based application designed to revolutionize traditional bookstore operations by providing an efficient, user-friendly digital platform. The system caters to both customers and administrators, offering seamless management of book inventory, ordering, and notifications. Customers can browse available books, place orders, and receive real-time notifications about stock availability. Administrators can manage inventory by adding new books, restocking existing ones, and receiving alerts for low or out-of-stock items. The project leverages key object-oriented programming (OOP) concepts such as encapsulation, inheritance, and polymorphism, ensuring a modular, maintainable, and scalable design. The Book Management Module handles adding, viewing, and updating inventory, while the Order Management Module processes customer orders and dynamically updates stock. A Notification Module provides real-time alerts, enhancing the user experience by keeping customers informed of restocked items and notifying administrators about inventory shortages. To ensure robustness, the system incorporates exception handling to manage invalid inputs and prevent runtime errors. The use of dynamic data structures, such as Array, List, enables efficient management of book and order data, ensuring smooth performance even with large inventories. The project also focuses on scalability, with potential future enhancements including database integration, secure payment processing, and a graphical user interface (GUI) for enhanced usability.

Overall, the Online Bookstore Project effectively modernizes the traditional bookstore model by automating inventory and order management, improving operational efficiency, and enhancing customer satisfaction.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	POs MAPPED	PSOs MAPPED
1. Applies object-oriented programming principles to develop a scalable and maintainable Java application.	**1**	**2**
2. Identifies issues in traditional bookstores like manual inventory management and customer inconvenience.	**2**	**1**
3. Prepares for future enhancements, showing adaptability to evolving requirements.	**2**	**3**

Note: 1- Low, 2-Medium, 3- High

SUPERVISOR

HEAD OF THE DEPARTMENT

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The primary objective of this project is to develop a comprehensive online bookstore system that enhances the user experience by offering a streamlined and intuitive platform for browsing, selecting, and purchasing books. The system aims to facilitate secure user authentication, advanced book categorization, and real-time stock management to ensure customers have access to accurate inventory information. Additionally, it provides a seamless shopping cart and checkout process, supported by automated notifications for restocked items, to improve customer satisfaction. This project also focuses on creating an efficient and scalable solution for administrators to manage inventory, track orders, and analyze sales trends effectively.

1.2 Overview

The Online Bookstore System is designed as a user-centric platform to transform the way books are purchased and managed. This project combines modern programming techniques with an intuitive interface to provide a seamless experience for both customers and administrators. The system comprises key features such as user authentication, dynamic book categorization, real-time stock tracking, and a secure checkout process. Customers can explore a vast collection of books, add items to their cart, and complete their purchases efficiently. For administrators, the system offers tools for managing inventory, monitoring sales, and ensuring operational efficiency.

1.3 Java Programming Concepts

The Online Bookstore project incorporates several core Java programming concepts to ensure efficiency and functionality. It utilizes encapsulation to protect data within classes and manage access through methods. Inheritance allows for code reuse, especially in managing user and book functionalities. Polymorphism is applied to enable dynamic behavior through method overriding, while abstraction simplifies complex operations by separating implementation details. The Collections Framework helps in managing dynamic data, such as books and user details, using structures like Array List. Exception handling ensures smooth program flow by catching errors, and Swing components facilitate a responsive GUI. Additionally, static methods support utility functions like notifications, and modular programming enhances code organization and scalability. These concepts work together to deliver a robust, scalable, and user-friendly online bookstore system.

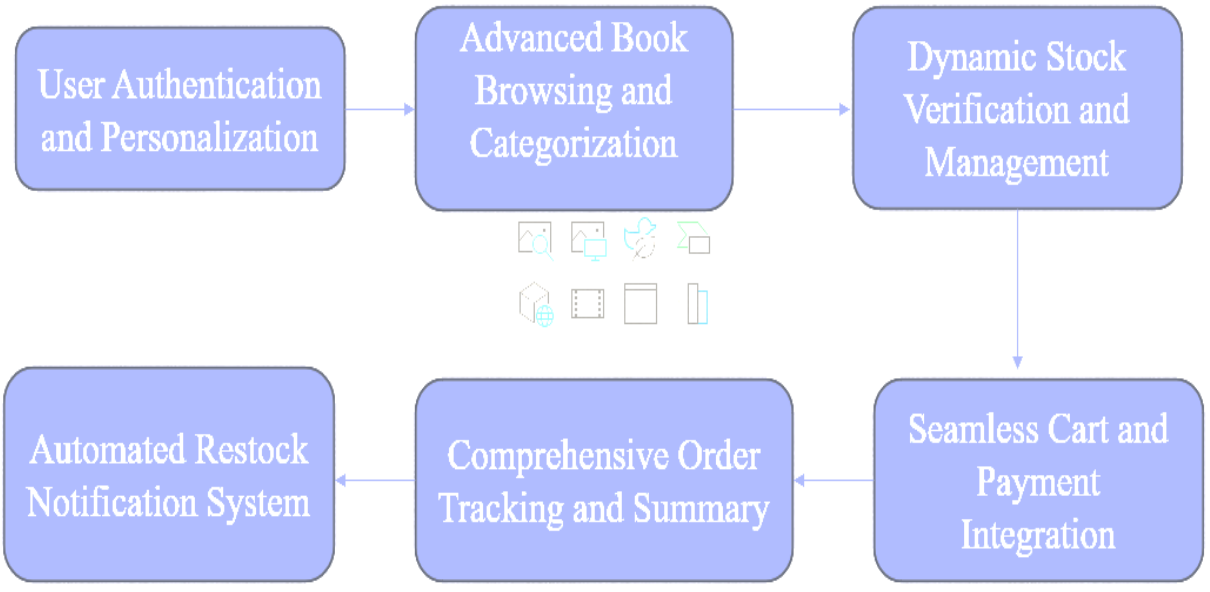
CHAPTER 2

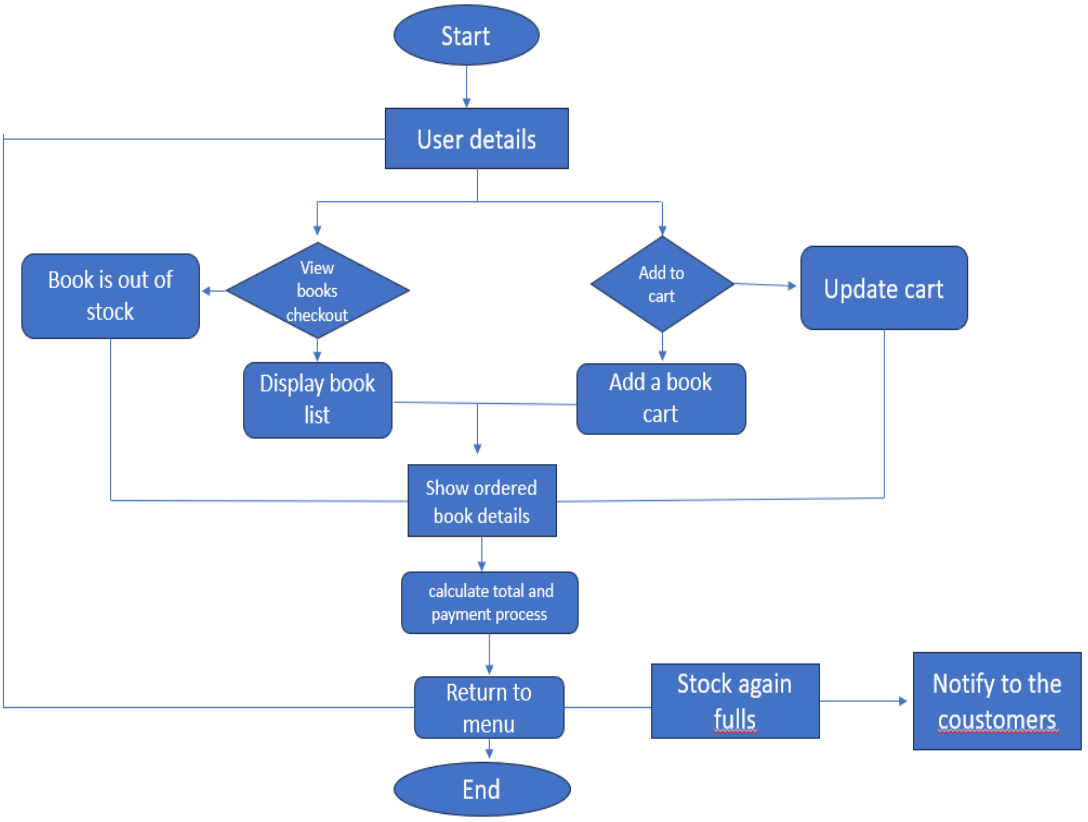
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work of the Online Bookstore project focuses on creating a comprehensive and efficient system to digitize the traditional book shopping experience. The project aims to implement a secure user authentication system to manage customer and administrator roles effectively, ensuring appropriate access to functionalities. Dynamic book categorization and an advanced search mechanism will be introduced to enable users to browse and find books easily, based on attributes like genre, author, or popularity. Real-time inventory tracking will ensure accurate stock information, with automated notifications for out-of-stock and restocked books to enhance user experience and streamline inventory management.

2.2 Block Diagram




Start
User details
Book is out of stock
View books checkout
Add to cart
Update cart
Display book list
Add a book cart
Show ordered book details
calculate total and payment process
Return to menu
Stock again fulls
Notify to the coustomers
End

CHAPTER 3

MODULE DESCRIPTION

3.1 User module

The User Module manages customer and administrator roles, handling login, registration, and personalization features. It ensures secure access to the system, allowing customers to browse books, make purchases, and track orders. Administrators can access additional functionalities such as inventory management and order processing. This module is responsible for maintaining user profiles and managing their interactions with the system.

3.2 Book Management Module

The Book Management Module handles all book-related operations within the system. It allows administrators to add new books, update existing book details, and manage inventory levels. This module ensures that the book catalog is up to date, enabling customers to search and browse books effectively. It also plays a vital role in managing stock levels and sending notifications when books are out of stock or restocked.

3.3 Order Management Module

The Order Management Module is responsible for processing customer orders, calculating totals, and updating stock levels in real-time. It tracks the status of orders, ensuring that customers receive timely updates on their purchase progress. This module integrates with the cart system to facilitate smooth checkout and payment processing, and it manages all aspects of order history and tracking for both customers and administrators

3.4 Notification Module

The Notification Module sends real-time alerts to both customers and administrators. Customers are notified when items they are interested in are restocked, while administrators receive alerts about low stock levels or out-of-stock items. This module ensures that both parties are kept informed and can take appropriate actions, improving user experience and operational efficiency.

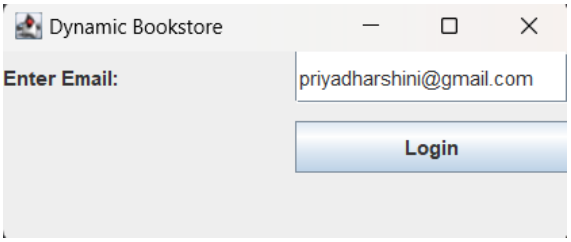
3.5 Stock Management Module

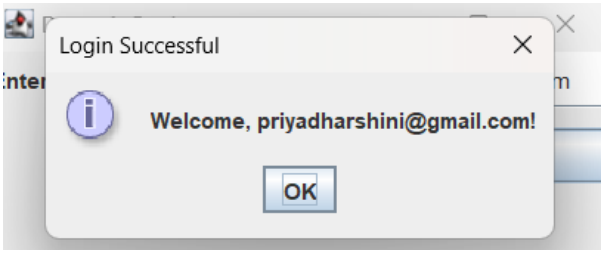
The Stock Management Module is crucial for tracking inventory levels in real-time. It prevents overstocking, adjusts stock based on customer orders, and triggers notifications when stock is low. This module ensures that customers can only order available items and helps administrators monitor and manage inventory effectively, preventing stockouts and overstock situations.

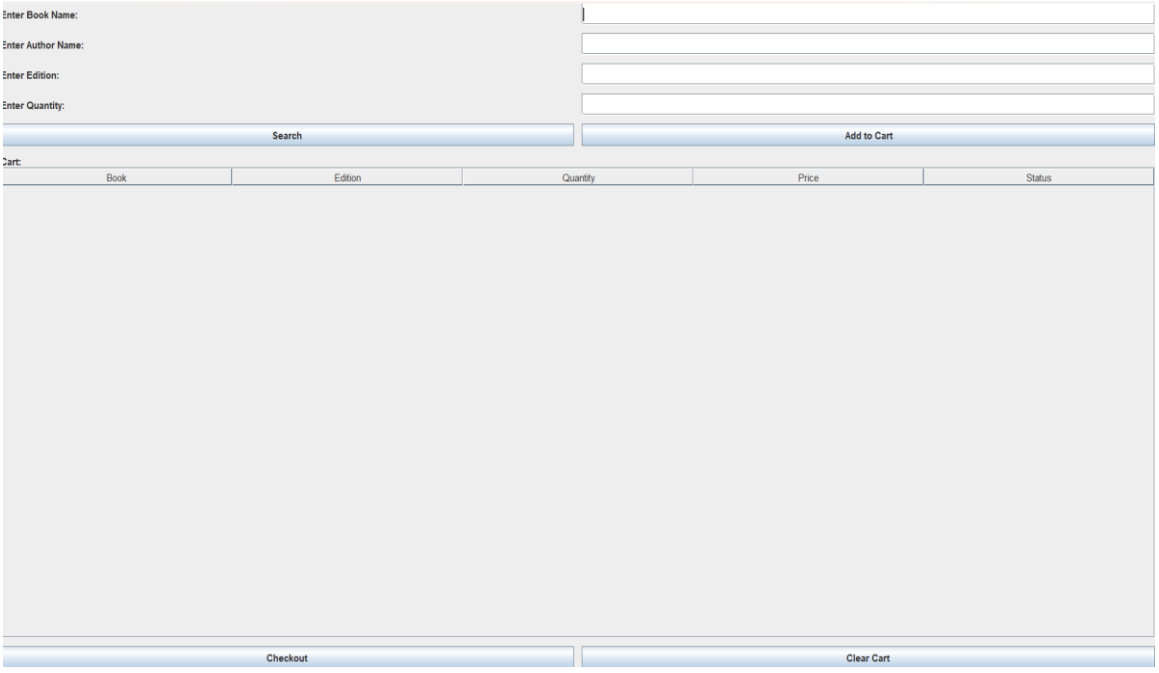
CHAPTER 4

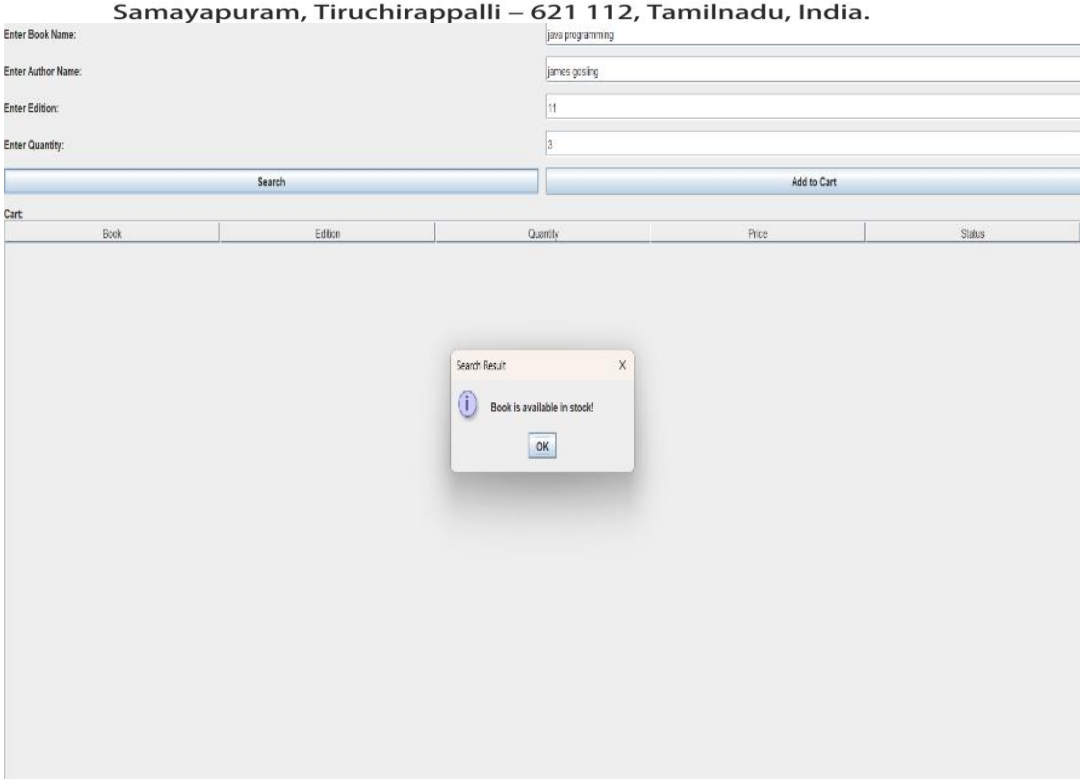
RESULTS AND DISCUSSION

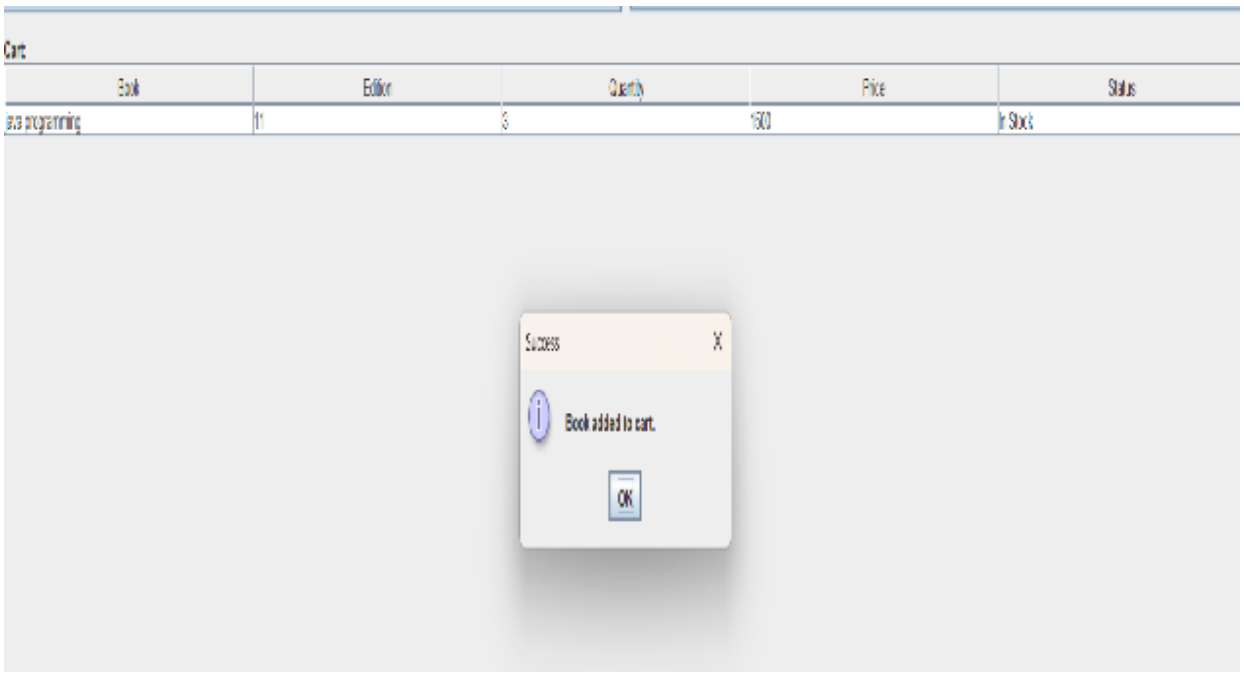
The Project successfully digitizes the book shopping and management process, providing seamless customer and admin functionalities. It ensures real-time stock updates, dynamic notifications, and a scalable design for future enhancements.

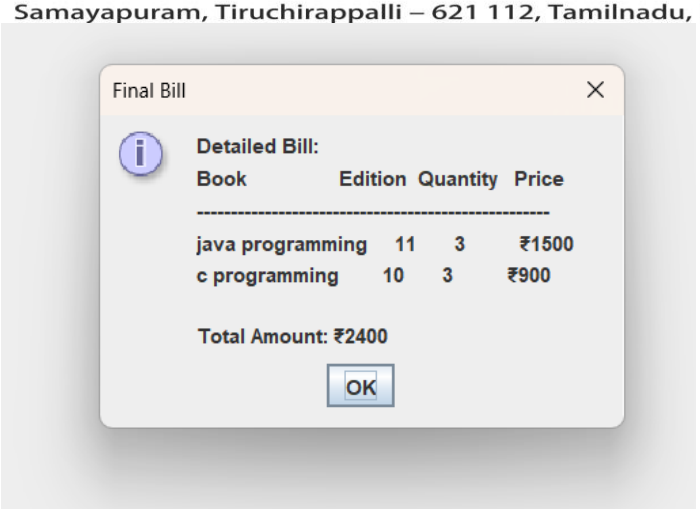


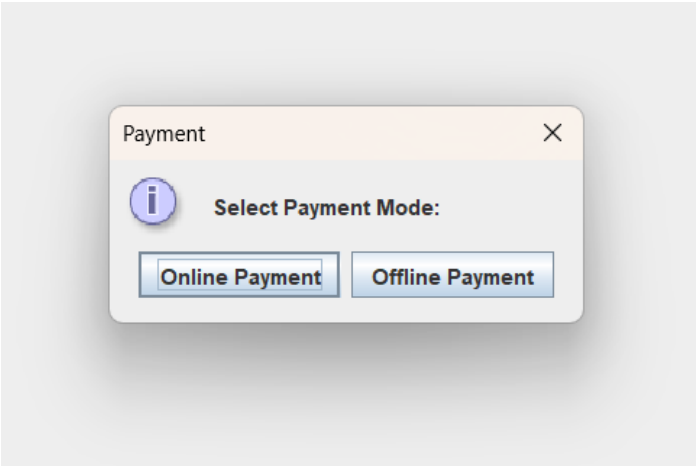


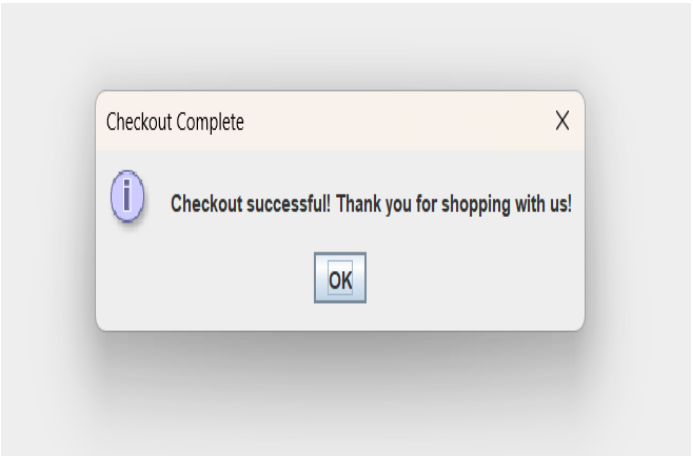



Enter Book Name:
java programming
Enter Author Name:
james gosling
Enter Edition:
11
Enter Quantity:
3
Search
Add to Cart
Cart:
Book
Edition
Quantity
Price
Status
Search Result
Book is available in stock!
OK


Cart:
Book
Edition
Quantity
Price
Status
11
3
500
In Stock
Success
Book added to cart.
OK


Final Bill
Detailed Bill:
Book Edition Quantity Price
java programming 11 3 ₹1500
c programming 10 3 ₹900
Total Amount: ₹2400
OK


Payment
Select Payment Mode:
Online Payment
Offline Payment


Checkout Complete
Checkout successful! Thank you for shopping with us!
OK

CHAPTER 5

CONCLUSION

The Online Bookstore project successfully achieves its goal of digitizing the traditional book shopping experience by offering a comprehensive, user-friendly platform for both customers and administrators. Through the seamless integration of core modules such as user authentication, book management, order processing, and inventory tracking, the system ensures a smooth and efficient browsing and purchasing experience. The real-time stock management and automated notification system further enhance the user experience by keeping both customers and administrators informed about inventory status.

REFERENCES

E-Commerce and Online Retail

- Chaffey, D. (2020). *E-Business and E-Commerce Management: Strategy, Implementation and Practice (7th Edition)*. Pearson Education.

- Laudon, K. C., & Traver, C. G. (2021). *E-Commerce 2021: Business, Technology, Society (16th Edition)*. Pearson.

You tube link

https://www.youtube.com/watch?v=OFb1pLLEkq4

https://www.youtube.com/watch?v=xYaPYqchKv4

https://www.youtube.com/watch?v=O7FOOU8kFSQ

APPENDIX

(Coding)

```
import javax.swing.*;
import java.awt.*;
import java.util.HashMap;
import java.util.Map;
import javax.swing.table.DefaultTableModel;

public class DynamicBookStore {

    static Map<String, Integer> bookStock = new HashMap<>();
    static Map<String, Integer> bookPrices = new HashMap<>();
    static Map<String, String> bookAuthors = new HashMap<>();
    static String userEmail = "";
    static DefaultTableModel cartTableModel;

    public static void main(String[] args) {
        initializeBooks();

        JFrame mainFrame = new JFrame("Dynamic Bookstore");
        mainFrame.setSize(400, 400);
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel loginPanel = new JPanel(new GridLayout(3, 2, 10, 10));
        JLabel emailLabel = new JLabel("Enter Email:");
        JTextField emailField = new JTextField();
```

```
JButton loginButton = new JButton("Login");
    loginPanel.add(emailLabel);
     loginPanel.add(emailField);
     loginPanel.add(new JLabel());
     loginPanel.add(loginButton);

     JPanel bookstorePanel = new JPanel(new BorderLayout(10, 10));
     JPanel inputPanel = new JPanel(new GridLayout(5, 2, 10, 10));

     JLabel bookNameLabel = new JLabel("Enter Book Name:");
     JTextField bookNameField = new JTextField();
     JLabel authorLabel = new JLabel("Enter Author Name:");
     JTextField authorField = new JTextField();
     JLabel editionLabel = new JLabel("Enter Edition:");
     JTextField editionField = new JTextField();
     JLabel quantityLabel = new JLabel("Enter Quantity:");
     JTextField quantityField = new JTextField();
     JButton searchButton = new JButton("Search");
     JButton addToCartButton = new JButton("Add to Cart");
     inputPanel.add(bookNameLabel);
     inputPanel.add(bookNameField);
     inputPanel.add(authorLabel);
     inputPanel.add(authorField);
     inputPanel.add(editionLabel);
     inputPanel.add(editionField);
     inputPanel.add(quantityLabel);
     inputPanel.add(quantityField);
     inputPanel.add(searchButton);
     inputPanel.add(addToCartButton);
     JPanel cartPanel = new JPanel(new BorderLayout());
     JLabel cartLabel = new JLabel("Cart:");
```

```
cartTableModel = new DefaultTableModel(new Object[]{"Book", "Edition",
"Quantity", "Price", "Status"}, 0);
        JTable cartTable = new JTable(cartTableModel);
        JScrollPane cartScrollPane = new JScrollPane(cartTable);
        cartPanel.add(cartLabel, BorderLayout.NORTH);
        cartPanel.add(cartScrollPane, BorderLayout.CENTER);

        JPanel checkoutPanel = new JPanel(new GridLayout(1, 2, 10, 10));
        JButton checkoutButton = new JButton("Checkout");
        JButton clearCartButton = new JButton("Clear Cart");

        checkoutPanel.add(checkoutButton);
        checkoutPanel.add(clearCartButton);

        bookstorePanel.add(inputPanel, BorderLayout.NORTH);
        bookstorePanel.add(cartPanel, BorderLayout.CENTER);
        bookstorePanel.add(checkoutPanel, BorderLayout.SOUTH);

        CardLayout cardLayout = new CardLayout();
        JPanel mainPanel = new JPanel(cardLayout);
        mainPanel.add(loginPanel, "Login");
        mainPanel.add(bookstorePanel, "Bookstore");

        mainFrame.add(mainPanel);
        final int[] totalAmount = {0};
```

```
        loginButton.addActionListener(e -> {
            userEmail = emailField.getText().trim();
            if (!userEmail.isEmpty() && userEmail.contains("@")) {
                JOptionPane.showMessageDialog(mainFrame, "Welcome, " + userEmail
+ "!", "Login Successful", JOptionPane.INFORMATION_MESSAGE);
                cardLayout.show(mainPanel, "Bookstore");
            } else {
                JOptionPane.showMessageDialog(mainFrame, "Please enter a valid
email!", "Error", JOptionPane.ERROR_MESSAGE);
            }
        });
    searchButton.addActionListener(e -> {
            String bookName = bookNameField.getText().trim().toLowerCase();
            String author = authorField.getText().trim();
            String edition = editionField.getText().trim();
            String bookKey = bookName + " edition " + edition;
            if (bookStock.containsKey(bookKey) &&
bookAuthors.get(bookKey).equalsIgnoreCase(author)) {
                JOptionPane.showMessageDialog(mainFrame, "Book is available in
stock!", "Search Result", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(mainFrame, "Book not found or author
mismatch.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        });
```

```
    addToCartButton.addActionListener(e -> {
        try {
            String bookName = bookNameField.getText().trim().toLowerCase();
            String author = authorField.getText().trim();
            String edition = editionField.getText().trim();
            int quantity = Integer.parseInt(quantityField.getText().trim());

            if (quantity <= 0) {
                throw new IllegalArgumentException("Quantity must be a positive
number.");
            }

            String bookKey = bookName + " edition " + edition;
            if (!edition.matches("\\d+") || Integer.parseInt(edition) < 1 ||
Integer.parseInt(edition) > 20) {
                throw new IllegalArgumentException("Please enter a valid edition.");
            }

            if (bookStock.containsKey(bookKey) &&
bookAuthors.get(bookKey).equalsIgnoreCase(author)) {
                int availableStock = bookStock.get(bookKey);
                String status = availableStock >= quantity ? "In Stock" : "Out of Stock";
                int price = bookPrices.get(bookKey) * quantity;
                cartTableModel.addRow(new Object[]{bookName, edition, quantity,
price, status});
```

```
 if (availableStock >= quantity) {
                    bookStock.put(bookKey, availableStock - quantity);
                    totalAmount[0] += price;
                    JOptionPane.showMessageDialog(mainFrame, "Book added to
cart.", "Success", JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(mainFrame, "Book added to cart
but quantity exceeds stock.", "Warning", JOptionPane.WARNING_MESSAGE);
                }
            } else {
                JOptionPane.showMessageDialog(mainFrame, "Book not found or
author mismatch.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(mainFrame, "Please enter valid
numbers for quantity and edition.", "Error", JOptionPane.ERROR_MESSAGE);
        } catch (IllegalArgumentException ex) {
            JOptionPane.showMessageDialog(mainFrame, ex.getMessage(), "Error",
JOptionPane.WARNING_MESSAGE);
        }
    });

    checkoutButton.addActionListener(e -> {
        if (cartTableModel.getRowCount() > 0) {
            StringBuilder billDetails = new StringBuilder("Detailed Bill:\n");
```

```
 billDetails.append(String.format("%-20s %-8s %-10s %-10s\n", "Book", "Edition",
"Quantity", "Price"));
            billDetails.append("-------------------------------------------------\n");

            for (int i = 0; i < cartTableModel.getRowCount(); i++) {
               billDetails.append(String.format("%-20s %-8s %-10s ₹%-10s\n",
                     cartTableModel.getValueAt(i, 0),
                     cartTableModel.getValueAt(i, 1),
                     cartTableModel.getValueAt(i, 2),
                     cartTableModel.getValueAt(i, 3)));
            }
            billDetails.append("\nTotal Amount: ₹").append(totalAmount[0]);

            JOptionPane.showMessageDialog(mainFrame, billDetails.toString(),
"Final Bill", JOptionPane.INFORMATION_MESSAGE);
            String[] paymentModes = {"Online Payment", "Offline Payment"};
            int paymentChoice = JOptionPane.showOptionDialog(mainFrame, "Select
Payment Mode:", "Payment",
                  JOptionPane.DEFAULT_OPTION,
JOptionPane.INFORMATION_MESSAGE, null, paymentModes,
paymentModes[0]);
               if (paymentChoice == 0) {
               JOptionPane.showMessageDialog(mainFrame, "Redirecting to Online
Payment Portal...", "Online Payment", JOptionPane.INFORMATION_MESSAGE);
            } else if (paymentChoice == 1)
```

```
JOptionPane.showMessageDialog(mainFrame, "Please pay at the counter during
delivery.", "Offline Payment", JOptionPane.INFORMATION_MESSAGE);
                }

                JOptionPane.showMessageDialog(mainFrame, "Checkout successful!
Thank you for shopping with us!", "Checkout Complete",
JOptionPane.INFORMATION_MESSAGE);
                cartTableModel.setRowCount(0);
                totalAmount[0] = 0;
                cardLayout.show(mainPanel, "Login");
            } else {
                JOptionPane.showMessageDialog(mainFrame, "Your cart is empty!",
"Error", JOptionPane.WARNING_MESSAGE);
            }
        });

        clearCartButton.addActionListener(e -> {
            cartTableModel.setRowCount(0);
            totalAmount[0] = 0;
            JOptionPane.showMessageDialog(mainFrame, "Cart cleared!",
"Information", JOptionPane.INFORMATION_MESSAGE);
        });

        mainFrame.setVisible(true);
    }
```

```
    static void initializeBooks() {
        String[] predefinedBooks = {
                "Java Programming,James Gosling,500",
                "Python Basics,Guido van Rossum,400",
                "C Programming,Dennis Ritchie,300",
                "Data Structures,Mark Weiss,450",
                "Web Development,Tim Berners-Lee,600",
                "Machine Learning,Andrew Ng,700"
        };

        for (String book : predefinedBooks) {
            for (int edition = 1; edition <= 20; edition++) {
                String bookKey = book.split(",")[0].toLowerCase() + " edition " + edition;
                bookStock.put(bookKey, 10);
                bookPrices.put(bookKey, Integer.parseInt(book.split(",")[2]));
                bookAuthors.put(bookKey, book.split(",")[1]);
            }
        }
    }
}
```