



ONLINE BOOKSTORE



A PROJECT REPORT

Submitted by

MATHAN M(2303811710421092)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)**

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “ **ONLINE BOOKSTORE** ” is the bonafide work of **MATHAN M (2303811710421092)** carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram-621112.

CGB1201-JAVA PROGRAMMING
Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.),
SUPERVISOR
ASSISTANT PROFESSOR

SIGNATURE

Mrs.K.Valli Priyadharshini, M.E.,(Ph.D.),

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram-621112.

Submitted for the viva-voce examination held on ...03.12.2024...

CGB1201-JAVA PROGRAMMING
Mr.MADHUKRISHNAN A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

INTERNAL EXAMINER

CGB1201-JAVA PROGRAMMING
Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.),
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8104-DSEC, PERAMBUR

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**ONLINE BOOKSTORE** ” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING**.

Signature

A handwritten signature in black ink, appearing to read 'M - Mathan', written over a horizontal line.

MATHAN M

Place: Samayapuram

Date:03.12.2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mrs. K. VALLI PRIYADHARSHINI, M.E., (Ph.D.)**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

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 modeling 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.

ABSTRACT

The online book store is a digital platform that enables users to browse, purchase, and download books from the comfort of their homes. By leveraging e-commerce technology, these platforms provide a vast collection of books, ranging from academic resources and fiction novels to self-help and specialized topics, catering to diverse reader interests. Unlike traditional bookstores, online book stores offer 24/7 accessibility, personalized recommendations through algorithms, and flexible purchasing options, including digital formats such as eBooks and audiobooks. These features enhance user convenience and accessibility while reducing operational costs for sellers. Additionally, they incorporate advanced search functionalities, user reviews, and secure payment methods to improve the shopping experience. The integration of mobile apps and subscription models further boosts customer engagement, making online book stores an essential component of the modern literary landscape. This abstract explores the operational framework, benefits, and challenges of online book stores, as well as their impact on the publishing industry and reading habits in a digitally-driven world.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

| ABSTRACT | POs MAPPED | PSOs MAPPED |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------|
| An online book store is a digital platform designed to facilitate the buying and selling of books through the internet. With the advent of e-commerce, online book stores have revolutionized the way readers access literature by providing a convenient, user-friendly, and diverse marketplace. Additionally, they incorporate advanced search functionalities, user reviews, and secure payment methods to improve the shopping experience. | PO1 -3 PO2 -3 PO3 -3 PO4 -3 PO5 -3 PO6 -3 PO7 -3 PO8 -3 PO9 -3 PO10 -3 PO11-3 PO12 -3 | PSO1 -3 PSO2 -3 PSO3 -3 |

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

TABLE OF CONTENTS

| CHAPTER NO | TITLE | PAGE NO. |
|------------|--------------------------------------|-------------|
| | ABSTRACT | viii |
| 1 | INTRODUCTION | 1 |
| | 1.1 Objective | 1 |
| | 1.2 Overview | 1 |
| | 1.3 Java Programming Concepts | 2 |
| | | 3 |
| 2 | PROJECT METHODOLOGY | 3 |
| | 2.1 Proposed Work | 3 |
| | 2.2 Block Diagram | 3 |
| 3 | MODULE DESCRIPTION | 4 |
| | 3.1 Book Management | 4 |
| | 3.2 Member Management | 4 |
| | 3.3 Booking the books | 4 |
| | 3.4 Search Capabilities | 4 |
| 4 | CONCLUSION & FUTURE SCOPE | 5 |
| | 4.1 Conclusion | 5 |
| | 4.2 Future Scope | 5 |
| | | 6 |
| | REFERENCES | 6 |
| | APPENDIX A(SOURCE CODE) | 7 |
| | APPENDIX B(SCREENSHOTS) | 16 |

CHAPTER 1

INTRODUCTION

1.1 Objective

The main objective of developing an Online Book Store is to create a comprehensive digital platform that facilitates the buying and selling of books, providing users with a seamless shopping experience. The system should be designed to handle a wide range of functionalities, including user registration, book catalog management, order processing, and payment handling. Ensuring that users can easily browse through various book categories, search for specific titles, and filter by author, genre, or price is critical to enhancing user engagement. Additionally, the platform should allow for real-time stock updates, ensuring customers have access to the most current availability of books. The project should also include features for customer support, such as contact forms, FAQs, and support chat functions, to help users with any questions or issues. The platform should be responsive and accessible, ensuring a positive experience across various devices like desktop computers, tablets, and smartphones. Ultimately, the objective is to build a user-friendly, efficient, and secure online book store that caters to the needs of book lovers, authors, and publishers, boosting accessibility and convenience in the book-buying process.

1.2 Overview

An Online Book Store is a digital platform that allows users to browse, purchase, and manage books from anywhere. It offers a wider selection compared to physical stores and provides convenient features like search filters, personalized recommendations, and user reviews. Secure payment options and user accounts enhance the shopping experience, while inventory management helps store owners keep track of stock. Overall, online book stores make book shopping more accessible and efficient, connecting readers with a broad range of titles and authors. Overall, an online book store combines convenience, accessibility, and functionality to cater to the evolving needs of modern readers, helping authors and publishers reach wider audiences while providing consumers with an engaging and personalized shopping experience.

1.3 Java Programming Concepts

The basic concepts of **Object-Oriented Programming (OOP)** are:

- ✓ **Class and Object:** A class is a blueprint, and an object is an instance of the class.
- ✓ **Encapsulation:** Bundles data and methods into a single unit (class) while restricting direct access to data.
- ✓ **Inheritance:** Enables a class (child) to inherit properties and methods from another class (parent), promoting code reuse.
- ✓ **Looping:** The while loop is used to create a menu-driven program that continuously runs until the user opts to exit.
- ✓ **Exception-Handling:** The try-catch block is used for error handling when parsing integers, ensuring that invalid input is managed gracefully.

Project related concepts

1. Classes and Objects

- ✓ **Classes and Objects:** The code is structured around classes like Book and Member that represent real-world entities and use object-oriented principles to model data and behaviors.

2. Encapsulation

- ✓ The Book and Member classes encapsulate their data (fields like id, title, author etc.) and provide methods to access and modify this data while hiding the internal implementation.

3. Methods

- ✓ Methods like `addBook()` and `viewBooks()` define the behavior of the `Book` class, and methods like `getName()` and `toString()` represents the member's details including their book history

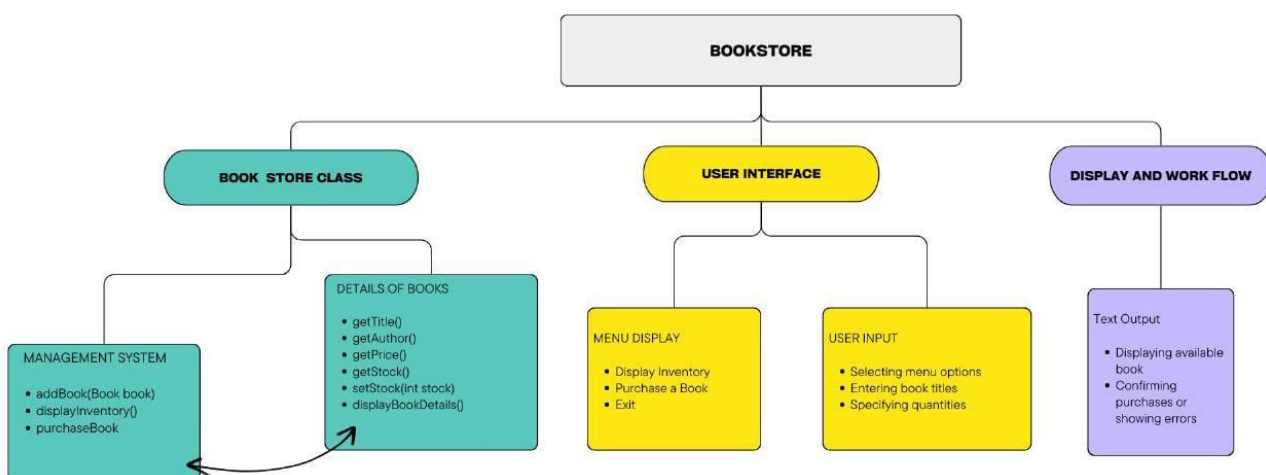
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The project proposes a comprehensive foundation for an online book store with key functionalities. It allows for effective book management by enabling users to add new books with details such as book ID, title, author, genre, and the number of available copies, as well as view all the books in the system and search for them by title, author, or genre. The member management feature supports adding members with details like member ID, name, contact number, and email, and provides a way to view all members and their issued books. For book issuing and returning, the system ensures that books can only be issued if they exist, the member is valid, and there are available copies, while returning books updates the system and tracks the changes in the member's issued list. The search functionalities allow users to look for books based on title, author, or genre and search for members by name or contact number. The code employs `HashMap` for efficient storage and retrieval of books, members, and issued book records, and uses `ArrayList` to manage the list of books issued to each member.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 Book management

The code implements robust book management for an online book store. Users can add books through the `addBook()` method, which checks for duplicates and validates inputs. The `viewBooks()` method displays all stored books or notifies if none are available, while `searchBooks()` enables users to find books by title, author, or genre. The `Book` class includes attributes like ID, title, author, genre, and available copies, with methods to access and modify these attributes.

3.2 Member management

It manages booking details through the `issuedBooks` map within the `Library` class, which records the relationship between book IDs and member IDs to track which member has borrowed which book. When a book is issued using the `issueBook()` method, it checks for the book's availability, verifies the member's existence, and updates the `issuedBooks` map along with the member's record to reflect the new borrowing. And it also shows the options of removing the entry, increments the available copies of the books.

3.3 Booking the books

Members are stored using a `HashMap` with unique member IDs, and each member is represented by an instance of the `Member` class. The `addMember()` method allows the addition of new members, ensuring uniqueness of member IDs and validating input data like name, contact number, and email. The `viewMembers()` method displays all members or informs if none exist.

3.4 Search capabilities

The code includes robust search capabilities to help users find books and members efficiently. The `searchBooks()` method allows searching by title, author, or genre, enabling users to input a keyword and find matching books in the library collection. It loops through the `books` map and displays books that meet the search criteria, providing flexibility in finding books based on different attributes. Similarly, the `searchMembers()` method enables searching for members by name or contact number.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

In conclusion, the online book store system developed in this code provides a comprehensive solution for managing books, members, and book transactions. With features like adding and viewing books and members, issuing and returning books, and powerful search capabilities, the system offers efficient management and user interaction. The implementation demonstrates the use of core Java concepts such as classes, collections, methods, and user input handling to create an interactive and functional application. This code serves as a solid foundation for building more complex book management systems or enhancing it with additional features like payment processing and user authentication. The system is modular, with clear methods dedicated to specific tasks, contributing to a well-structured and maintainable codebase. This design makes it easy to scale the project with additional features like online payment integration, user authentication, or real-time stock updates. Overall, this online book store system provides a strong foundation for digital library management, offering essential functionalities that can be expanded and adapted for more complex needs.

4.2 FUTURE SCOPE

The future scope of the online book store system is vast and full of potential for growth and enhancement. Enhancements such as robust user authentication systems can improve data security and user management, while integrating payment gateways would allow users to make direct purchases, transforming the system into a full-fledged online store. The search functionality could be upgraded with more filters, like price range, publication year, and user ratings, to create a more personalized shopping experience. Adding a recommendation system based on user history would further engage customers and boost sales. Developing a mobile application would expand accessibility, allowing users to shop on smartphones and tablets. Additionally, implementing real-time inventory management, stock alerts, and automated reordering processes would help maintain optimal stock levels.

REFERENCES

Java Books:

1. "Java:The Complete Reference" by Herbert Schildt

This book is a great resource for beginners learning Java, with a focus on object-oriented programming concepts and real-world application development.

2. "Effective Java" by Joshua Bloch

A deeper dive into best practices for writing clean, maintainable Java code. It covers advanced topics like Java collections, concurrency, and design patterns that could be applied to more complex payroll systems.

Websites:

1. GeeksforGeeks - Java Tutorials

- URL: <https://www.geeksforgeeks.org/java/>
- A comprehensive collection of tutorials on Java, covering topics like classes, objects, inheritance, encapsulation, and more. Great for learning the core concepts of Java and applying them in projects like Online Book Store.

2. W3Schools - Java Tutorial

- URL: <https://www.w3schools.com/java/>
- A beginner-friendly resource that offers tutorials on Java programming, including object-oriented principles and core Java concepts.

3. JavaTpoint - Java Tutorial

- URL: <https://www.javatpoint.com/>
- Offers Java tutorials from the basics to advanced concepts. And provides detailed guides on Java programming, including working with objects and classes, which are crucial for building a Online Book Store.

APPENDIX A (SOURCE CODE)

```
import javax.swing.*;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.util.ArrayList;

class Book {
    private String title;
    private String author;
    private double price;
    private int stock;

    public Book(String title, String author, double price, int stock) {
        this.title = title;
        this.author = author;
        this.price = price;
        this.stock = stock;
    }

    public String getTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }

    public double getPrice() {
        return price;
    }

    public int getStock() {
        return stock;
    }

    public void setStock(int stock) {
        this.stock = stock;
    }
}

class Inventory {
    private ArrayList<Book> books;

    public Inventory() {
        books = new ArrayList<>();
    }

    public void addBook(Book book) {
        books.add(book);
    }
}
```

```

    public ArrayList<Book> getBooks() {
        return books;
    }
}

public class Main extends JFrame {
    private CardLayout cardLayout;
    private JPanel mainPanel;
    private Inventory inventory;
    private String registeredUsername;
    private String registeredPassword;
    private JTextArea cartArea;
    private JTextArea trackerArea;
    private DefaultTableModel tableModel;
    private JTable table;
    private double total;

    public Main() {
        inventory = new Inventory();
        setupInventory();

        // Setting up the JFrame
        setTitle("Online Bookstore");
        setSize(800, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Card Layout for navigation
        cardLayout = new CardLayout();
        mainPanel = new JPanel(cardLayout);

        // Adding Pages
        mainPanel.add(createRegistrationPage(), "Register");
        mainPanel.add(createLoginPage(), "Login");
        mainPanel.add(createHomePage(), "Home");
        mainPanel.add(createBookstorePage(), "Bookstore");
        mainPanel.add(createBookTrackerPage(), "BookTracker");

        add(mainPanel);
        setVisible(true);

        // Start at Registration Page
        cardLayout.show(mainPanel, "Register");
    }

    private JPanel createRegistrationPage() {
        JPanel registerPanel = new JPanel(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();

        JLabel usernameLabel = new JLabel("Set Username:");
        JLabel passwordLabel = new JLabel("Set Password:");
        JTextField usernameField = new JTextField(15);
        JPasswordField passwordField = new JPasswordField(15);

```

```

JLabel strengthLabel = new JLabel("Password Strength: ");
JButton registerButton = new JButton("Register");

// Labels and boxes for password conditions
JLabel conditionLowercase = new JLabel("Contains lowercase:");
JLabel conditionUppercase = new JLabel("Contains uppercase:");
JLabel conditionNumber = new JLabel("Contains number:");
JLabel conditionLength = new JLabel("At least 8 characters:");
JCheckBox boxLowercase = new JCheckBox();
JCheckBox boxUppercase = new JCheckBox();
JCheckBox boxNumber = new JCheckBox();
JCheckBox boxLength = new JCheckBox();

// Disable checkboxes to make them display-only
boxLowercase.setEnabled(false);
boxUppercase.setEnabled(false);
boxNumber.setEnabled(false);
boxLength.setEnabled(false);

// Add listener to check password strength and update boxes
passwordField.getDocument().addDocumentListener(new DocumentListener() {
    @Override
    public void insertUpdate(DocumentEvent e) {
        updatePasswordStrength();
    }

    @Override
    public void removeUpdate(DocumentEvent e) {
        updatePasswordStrength();
    }

    @Override
    public void changedUpdate(DocumentEvent e) {
        updatePasswordStrength();
    }

    private void updatePasswordStrength() {
        String password = new String(passwordField.getPassword());

        // Update boxes based on the password
        boxLowercase.setSelected(password.chars().anyMatch(Character::isLowerCase));
        boxUppercase.setSelected(password.chars().anyMatch(Character::isUpperCase));
        boxNumber.setSelected(password.chars().anyMatch(Character::isDigit));
        boxLength.setSelected(password.length() >= 8);

        // Update overall strength label
        String strength = validatePassword(password);
        strengthLabel.setText("Password Strength: " + strength);
    }
});

registerButton.addActionListener(e -> {
    registeredUsername = usernameField.getText();

```

```

        registeredPassword = new String(passwordField.getPassword());

        if (!registeredUsername.isEmpty() && !registeredPassword.isEmpty() &&
            validatePassword(registeredPassword).startsWith("Strong")) {
            JOptionPane.showMessageDialog(this, "Registration Successful!", "Success",
                JOptionPane.INFORMATION_MESSAGE);
            cardLayout.show(mainPanel, "Login");
        } else {
            JOptionPane.showMessageDialog(this, "Please set a strong password!", "Error",
                JOptionPane.ERROR_MESSAGE);
        }
    });

    // Layout components
    gbc.insets = new Insets(5, 5, 5, 5);

    gbc.gridx = 0;
    gbc.gridy = 0;
    registerPanel.add(usernameLabel, gbc);
    gbc.gridx = 1;
    registerPanel.add(usernameField, gbc);

    gbc.gridx = 0;
    gbc.gridy = 1;
    registerPanel.add(passwordLabel, gbc);
    gbc.gridx = 1;
    registerPanel.add(passwordField, gbc);

    gbc.gridx = 1;
    gbc.gridy = 2;
    registerPanel.add(strengthLabel, gbc);

    // Add password condition boxes and labels
    gbc.gridx = 0;
    gbc.gridy = 3;
    registerPanel.add(conditionLowercase, gbc);
    gbc.gridx = 1;
    registerPanel.add(boxLowercase, gbc);

    gbc.gridx = 0;
    gbc.gridy = 4;
    registerPanel.add(conditionUppercase, gbc);
    gbc.gridx = 1;
    registerPanel.add(boxUppercase, gbc);

    gbc.gridx = 0;
    gbc.gridy = 5;
    registerPanel.add(conditionNumber, gbc);
    gbc.gridx = 1;
    registerPanel.add(boxNumber, gbc);

    gbc.gridx = 0;
    gbc.gridy = 6;
    registerPanel.add(conditionLength, gbc);

```

```

gbc.gridx = 1;
registerPanel.add(boxLength, gbc);

gbc.gridx = 1;
gbc.gridy = 7;
registerPanel.add(registerButton, gbc);

return registerPanel;
}

private String validatePassword(String password) {
    if (password.length() < 3) {
        return "Weak 😞";
    } else if (password.length() >= 3 && password.length() < 8) {
        return "Medium 😊";
    } else {
        boolean hasUpper = password.chars().anyMatch(Character::isUpperCase);
        boolean hasLower = password.chars().anyMatch(Character::isLowerCase);
        boolean hasDigit = password.chars().anyMatch(Character::isDigit);
        boolean hasSpecial = password.chars().anyMatch(c -> "!@#$%^&*()_+[]{}|;:'.<>?/".indexOf(c)
        >= 0);

        if (hasUpper && hasLower && hasDigit && hasSpecial) {
            return "Strong 🛡️";
        } else {
            return "Medium 😊";
        }
    }
}

private JPanel createLoginPage() {
    JPanel loginPanel = new JPanel(new GridBagLayout());
    GridBagConstraints gbc = new GridBagConstraints();

    JLabel usernameLabel = new JLabel("Username:");
    JLabel passwordLabel = new JLabel("Password:");
    JTextField usernameField = new JTextField(15);
    JPasswordField passwordField = new JPasswordField(15);
    JButton loginButton = new JButton("Login");

    loginButton.addActionListener(e -> {
        String enteredUsername = usernameField.getText();
        String enteredPassword = new String(passwordField.getPassword());

        if (enteredUsername.equals(registeredUsername) &&
        enteredPassword.equals(registeredPassword)) {
            JOptionPane.showMessageDialog(this, "Login Successful!", "Success",
            JOptionPane.INFORMATION_MESSAGE);
            cardLayout.show(mainPanel, "Home");
        } else {
            JOptionPane.showMessageDialog(this, "Invalid credentials!", "Error",
            JOptionPane.ERROR_MESSAGE);
        }
    })
}

```

```

});

gbc.insets = new Insets(5, 5, 5, 5);
gbc.gridx = 0;
gbc.gridy = 0;
loginPanel.add(usernameLabel, gbc);
gbc.gridx = 1;
loginPanel.add(usernameField, gbc);

gbc.gridx = 0;
gbc.gridy = 1;
loginPanel.add(passwordLabel, gbc);
gbc.gridx = 1;
loginPanel.add(passwordField, gbc);

gbc.gridx = 1;
gbc.gridy = 2;
loginPanel.add(loginButton, gbc);

return loginPanel;
}

private JPanel createHomePage() {
    JPanel homePanel = new JPanel(new GridLayout(3, 1));
    JButton viewInventoryButton = new JButton("View Inventory");
    JButton bookTrackerButton = new JButton("Book Tracker");
    JButton exitButton = new JButton("Exit");

    viewInventoryButton.addActionListener(e -> cardLayout.show(mainPanel, "Bookstore"));
    bookTrackerButton.addActionListener(e -> cardLayout.show(mainPanel, "BookTracker"));
    exitButton.addActionListener(e -> {
        int option = JOptionPane.showConfirmDialog(this, "Are you sure you want to exit?", "Exit",
JOptionPane.YES_NO_OPTION);
        if (option == JOptionPane.YES_OPTION) {
            System.exit(0);
        }
    });

    homePanel.add(viewInventoryButton);
    homePanel.add(bookTrackerButton);
    homePanel.add(exitButton);
    return homePanel;
}

private JPanel createBookstorePage() {
    JPanel bookstorePanel = new JPanel(new BorderLayout());

    // Book Table
    tableModel = new DefaultTableModel(new String[]{"Title", "Author", "Price", "Stock"}, 0) {
        @Override
        public boolean isCellEditable(int row, int column) {
            return false; // Disables editing of table cells
        }
    }

```

```

};
table = new JTable(tableModel);
refreshTable();
bookstorePanel.add(new JScrollPane(table), BorderLayout.CENTER);

// Buttons Panel
JPanel buttonPanel = new JPanel();
JButton addToCartButton = new JButton("Add to Cart");
JButton checkoutButton = new JButton("Checkout");
JButton backButton = new JButton("Back");
buttonPanel.add(addToCartButton);
buttonPanel.add(checkoutButton);
buttonPanel.add(backButton);
bookstorePanel.add(buttonPanel, BorderLayout.SOUTH);

// Cart Area
cartArea = new JTextArea(10, 30);
cartArea.setEditable(false);
bookstorePanel.add(new JScrollPane(cartArea), BorderLayout.EAST);

addToCartButton.addActionListener(e -> addToCart());
checkoutButton.addActionListener(e -> checkout());
backButton.addActionListener(e -> cardLayout.show(mainPanel, "Home"));

return bookstorePanel;
}

private JPanel createBookTrackerPage() {
    JPanel trackerPanel = new JPanel(new BorderLayout());

    trackerArea = new JTextArea(10, 30);
    trackerArea.setEditable(false);
    trackerPanel.add(new JScrollPane(trackerArea), BorderLayout.CENTER);

    trackerArea.append("Book Tracking Status:\n");
    trackerArea.append("Manufactured in Delhi, Posted to Chennai\n");
    trackerArea.append("3 Days for Delivery\n");

    // Back button for Book Tracker
    JButton backButton = new JButton("Back");
    backButton.addActionListener(e -> cardLayout.show(mainPanel, "Home"));
    trackerPanel.add(backButton, BorderLayout.SOUTH);

    return trackerPanel;
}

private void setupInventory() {
    // Add more than 30 books to the inventory
    inventory.addBook(new Book("The Catcher in the Rye", "J.D. Salinger", 10.99, 5));
    inventory.addBook(new Book("To Kill a Mockingbird", "Harper Lee", 12.99, 3));
    inventory.addBook(new Book("1984", "George Orwell", 15.50, 4));

    // Add 30 more books here (for demonstration purposes, we'll just use some variations)

```

```

        for (int i = 4; i <= 34; i++) {
            inventory.addBook(new Book("Book Title " + i, "Author " + i, 10.00 + i, i));
        }
    }

    private void refreshTable() {
        tableModel.setRowCount(0);
        for (Book book : inventory.getBooks()) {
            tableModel.addRow(new Object[]{book.getTitle(), book.getAuthor(), book.getPrice(),
book.getStock()});
        }
    }

    private void addToCart() {
        int selectedRow = table.getSelectedRow();
        if (selectedRow != -1) {
            String title = table.getValueAt(selectedRow, 0).toString();
            double price = Double.parseDouble(table.getValueAt(selectedRow, 2).toString());
            int stock = Integer.parseInt(table.getValueAt(selectedRow, 3).toString());

            if (stock > 0) {
                String quantityStr = JOptionPane.showInputDialog(this, "Enter quantity:", "1");
                int quantity = 1;
                try {
                    quantity = Integer.parseInt(quantityStr);
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(this, "Invalid quantity entered!", "Error",
JOptionPane.ERROR_MESSAGE);
                }

                if (quantity <= stock) {
                    cartArea.append(title + " - $" + price + " x " + quantity + "\n");
                    total += price * quantity;
                    inventory.getBooks().get(selectedRow).setStock(stock - quantity);
                    refreshTable();
                } else {
                    JOptionPane.showMessageDialog(this, "Not enough stock!", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            } else {
                JOptionPane.showMessageDialog(this, "Out of Stock!", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog(this, "No book selected!", "Warning",
JOptionPane.WARNING_MESSAGE);
        }
    }

    private void checkout() {
        if (total > 0) {
            JOptionPane.showMessageDialog(this, "Total: $" + total, "Checkout",
JOptionPane.INFORMATION_MESSAGE);
            cartArea.setText("");
        }
    }

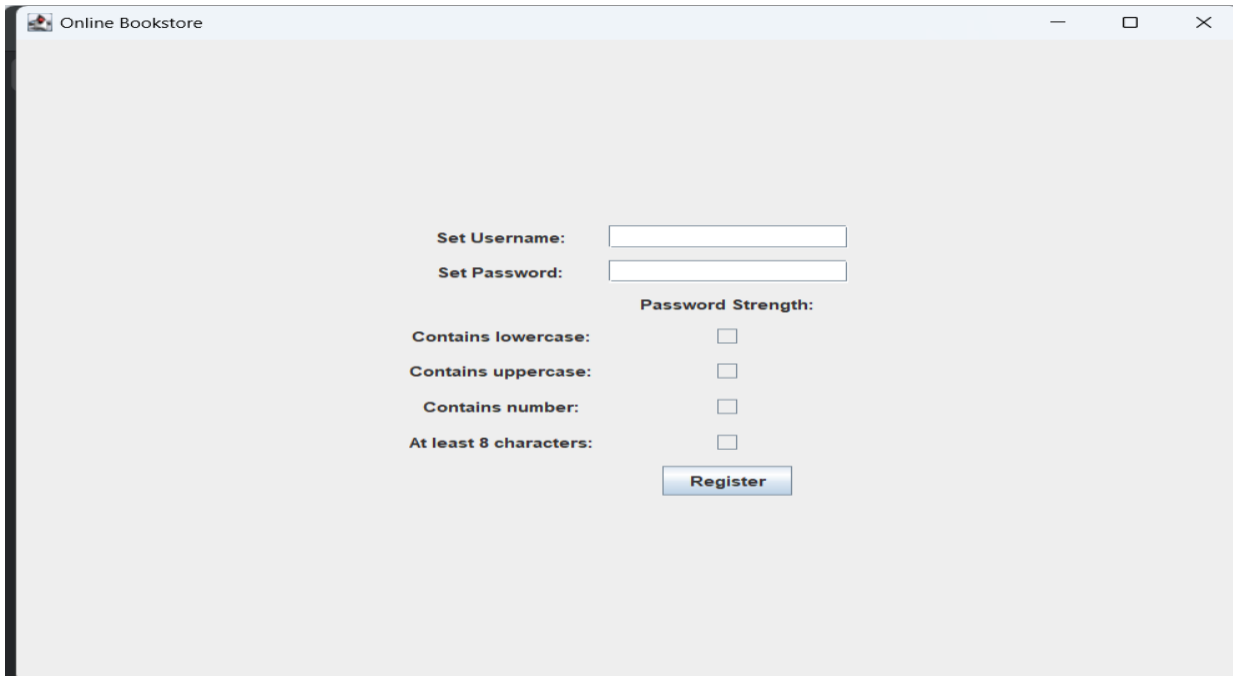
```



```
        total = 0;
    } else {
        JOptionPane.showMessageDialog(this, "Cart is empty!", "Warning",
JOptionPane.WARNING_MESSAGE);
    }
}

public static void main(String[] args) {
    new Main();
}
}
```

APPENDIX B(SCREENSHOTS)



The screenshot shows a window titled "Online Bookstore" with a registration form. The form includes two text input fields for "Set Username:" and "Set Password:". Below the password field is a "Password Strength:" section with four checkboxes: "Contains lowercase:", "Contains uppercase:", "Contains number:", and "At least 8 characters:". A "Register" button is positioned below the checkboxes.

Online Bookstore

Set Username:

Set Password:

Password Strength:

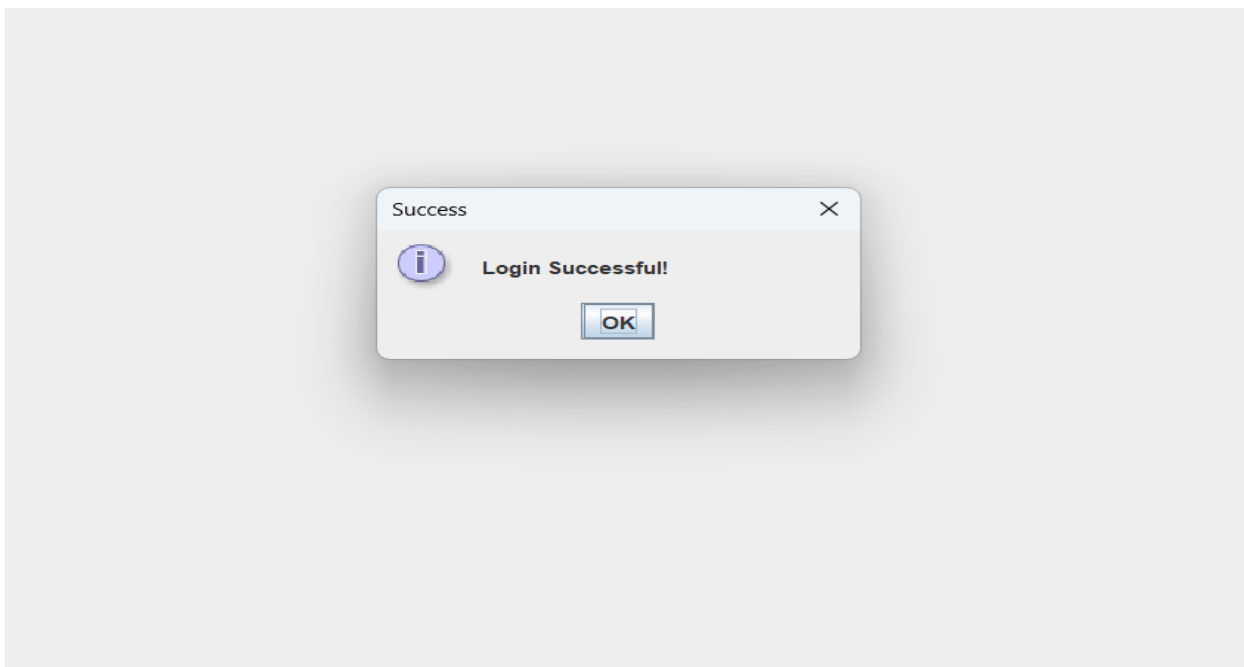
Contains lowercase: ☐

Contains uppercase: ☐

Contains number: ☐

At least 8 characters: ☐

Register



[View Inventory](#)

[Book Tracker](#)

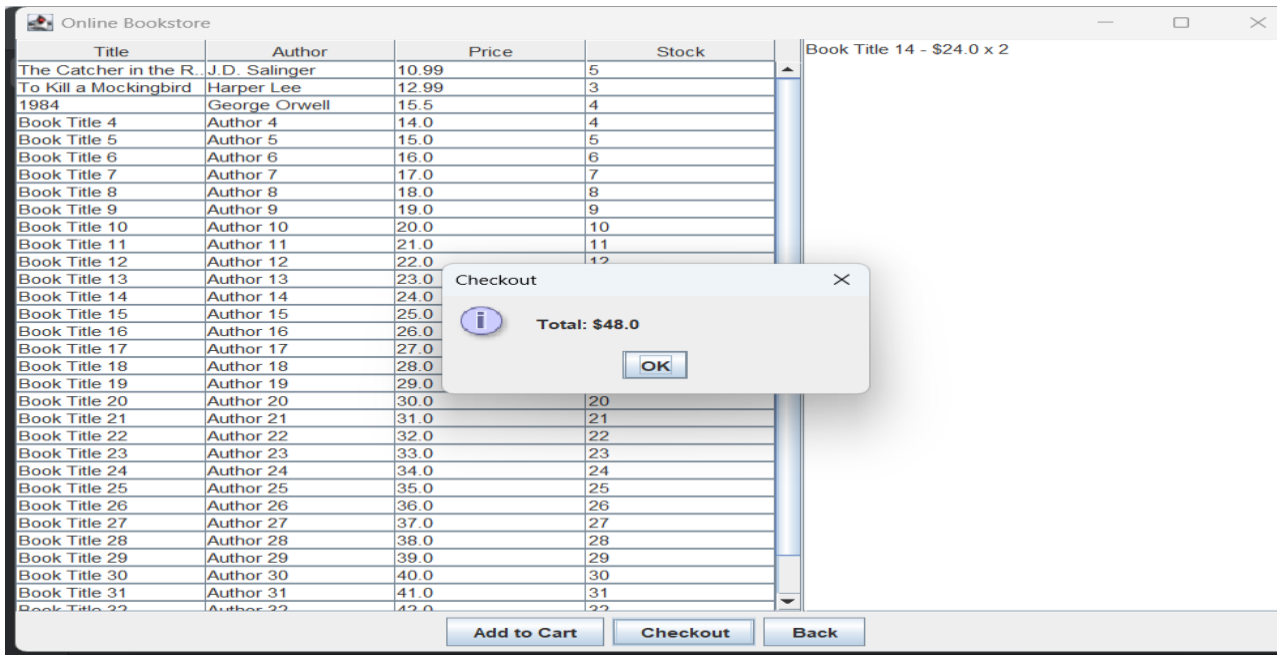
[Exit](#)

| Title | Author | Price | Stock |
|-------------------------|---------------|-------|-------|
| The Catcher in the R... | J.D. Salinger | 10.99 | 5 |
| To Kill a Mockingbird | Harper Lee | 12.99 | 3 |
| 1984 | George Orwell | 15.5 | 4 |
| Book Title 4 | Author 4 | 14.0 | 4 |
| Book Title 5 | Author 5 | 15.0 | 5 |
| Book Title 6 | Author 6 | 16.0 | 6 |
| Book Title 7 | Author 7 | 17.0 | 7 |
| Book Title 8 | Author 8 | 18.0 | 8 |
| Book Title 9 | Author 9 | 19.0 | 9 |
| Book Title 10 | Author 10 | 20.0 | 10 |
| Book Title 11 | Author 11 | 21.0 | 11 |
| Book Title 12 | Author 12 | 22.0 | 12 |
| Book Title 13 | Author 13 | 23.0 | 13 |
| Book Title 14 | Author 14 | 24.0 | 14 |
| Book Title 15 | Author 15 | 25.0 | 15 |
| Book Title 16 | Author 16 | 26.0 | 16 |
| Book Title 17 | Author 17 | 27.0 | 17 |
| Book Title 18 | Author 18 | 28.0 | 18 |
| Book Title 19 | Author 19 | 29.0 | 19 |
| Book Title 20 | Author 20 | 30.0 | 20 |
| Book Title 21 | Author 21 | 31.0 | 21 |
| Book Title 22 | Author 22 | 32.0 | 22 |
| Book Title 23 | Author 23 | 33.0 | 23 |
| Book Title 24 | Author 24 | 34.0 | 24 |
| Book Title 25 | Author 25 | 35.0 | 25 |
| Book Title 26 | Author 26 | 36.0 | 26 |
| Book Title 27 | Author 27 | 37.0 | 27 |
| Book Title 28 | Author 28 | 38.0 | 28 |
| Book Title 29 | Author 29 | 39.0 | 29 |
| Book Title 30 | Author 30 | 40.0 | 30 |
| Book Title 31 | Author 31 | 41.0 | 31 |
| Book Title 32 | Author 32 | 42.0 | 32 |

[Add to Cart](#)

[Checkout](#)

[Back](#)



Book Tracking Status:
Manufactured in Delhi, Posted to Chennai
3 Days for Delivery

Back

