A Project Report on

# ONLINE BOOK BUYING AND SELLING PORTAL

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

# EGB1201 – JAVA PROGRAMMING

Under the guidance of

## Mrs. P KAYALVIZHI M.E.,

**Assistant Professor / CSE**

Submitted By

**SWATHI D (927622BEC225)**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**M.KUMARASAMY COLLEGE OF ENGINEERING**

(Autonomous)

# KARUR – 639 113

DECEMBER 2024

i

# M. KUMARASAMY COLLEGE OF ENGINEERING

**(Autonomous Institution affiliated to Anna University, Chennai)**

# KARUR – 639 113

**BONAFIDE CERTIFICATE**

Certified that this project report on **“ONLINE BOOK BUYING AND SELLING PORTAL”** is the bonafide work of **D.SWATHI(927622BEC225)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

|  |  |
| --- | --- |
| Signature | Signature |
| **Mrs. P.KAYALVIZHI M.E.,** | **Dr. A. KAVITHA M.E.,Ph.D.,** |
| **SUPERVISOR,** | **HEAD OF THE DEPARTMENT,** |
| Department of Computer Science and Engineering, | Department of Electronics and Communication Engineering, |
| M.Kumarasamy College of Engineering, | M.Kumarasamy College of Engineering, |
| Thalavapalayam, Karur -639 113. | Thalavapalayam, Karur -639 113. |

# 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 empower the Electronics and Communication Engineering students with emerging technologies, professionalism, innovative research and social responsibility **MISSION OF THE DEPARTMENT**

* Attain the academic excellence through innovative teaching learning process, research areas & laboratories and Consultancy projects
* Inculcate the students in problem solving and lifelong learning ability
* Provide entrepreneurial skills and leadership qualities
* Render the technical knowledge and skills of faculty members Produce hi-tech professionals in the field of Electrical and Electronics Engineering by inculcating core knowledge.

## PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

**PEO 1: Core Competence:** Graduates will have a successful career in academia or industry associated with Electronics and Communication Engineering

**PEO 2: Professionalism:** Graduates will provide feasible solutions for the challenging problemsthroughcomprehensiveresearchandinnovationinthealliedareasofElectronic sandCommunicationEngineering.

**PEO 3: Lifelong Learning :**Graduates will contribute to the social needs through lifelong learning, practicing professional ethics and leadership quality.

## 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
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, 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:** Applying knowledge in various areas, like Electronics, Communications, Signal processing, VLSI, Embedded systems etc., in the design and implementation of Engineering application.

**PSO2:** Able to solve complex problems in Electronics and Communication Engineering with analytical and managerial skills either independently or in team using latest hardware and software tools to fulfil the industrial expectations.

# ABSTRACT

The "Online Book Portal" is an innovative web-based platform designed to simplify the buying and selling of books. This system enables users to log in, browse available books, add books to their cart, purchase books. Built with Java, the application integrates core programming principles such as object-oriented programming, data structures, and event-driven design. The project ensures user convenience, scalability, and secure interactions for a seamless book trading experience.

## ABSTRACT WITH POs AND PSOs MAPPING

|  |  |  |
| --- | --- | --- |
| **ABSTRACT** | **POs**  **MAPPED** | **PSOs**  **MAPPED** |
| The "Online Book Portal" is an innovative web-based platform designed to simplify the buying and selling of books. This system enables users to log in, browse available books, add books to their cart, purchase books. Built with Java, the application integrates core programming principles such as object-oriented programming, data structures, and event-driven design. The project ensures user convenience, scalability, and secure interactions for a seamless book trading experience | **PO1(3)**  **PO2(3)**  **PO3(3)**  **PO4(2)**  **PO5(3)**  **PO6(2)**  **PO7(2)**  **PO8(1)**  **PO9(2) PO10(2) PO11(2) PO12(2)** | **PSO1(1) PSO2(2)** |

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

**SUPERVISOR HEAD OF THE DEPARTMENT**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER**  **No.** | **TITLE** | **PAGE**  **No.** |
|  | **ABSTRACT** | **Vii** |
| **1** | **INTRODUCTION** | **1** |
|  | 1.1 Objective | 1 |
|  | 1.2 Overview | 1 |
|  | 1.3 Java Programming concepts | 1 |
| **2** | **PROJECT METHODOLOGY** | **3** |
|  | 2.1Proposed Work | 3 |
|  | 2.2 Block Diagram | 3 |
| **3** | **MODULE DESCRIPTION** | **4** |
|  | 3.1 User Interface (UI) Module | 4 |
|  | 3.2 Event Handling Module | 4 |
|  | 3.3 Book Management Module | 4 |
|  | 3.4 Book Actions Module | 5 |
|  | 3.5 Window Closing Module | 5 |
| **4** | **RESULTS AND DISCUSSION** | **6** |
| **5** | **CONCLUSION** | **8** |
|  | **REFERENCES** | **9** |
|  | **APPENDIX** | **10** |

**CHAPTER 1 INTRODUCTION**

## Objective

The objective of the online book buying and selling portal is to create a seamless platform for users to easily buy, sell, and exchange books. It aims to offer a wide variety of new and used books at competitive prices, catering to diverse interests. The portal will ensure secure transactions and a user-friendly experience. It will also provide convenient payment options and fast delivery services. The goal is to foster a community of book enthusiasts while promoting sustainable reading practices

## Overview

The online book buying and selling portal is a platform that connects buyers and sellers of books in a secure, user-friendly environment. It offers a wide variety of new and used books across different genres. Users can list their books for sale, browse available titles, and make secure transactions. The platform promotes sustainability by encouraging book reuse and fostering a community of readers. It also ensures fast delivery and responsive customer support for a seamless experience

# JAVA PROGRAMMING CONCEPTS :

* + 1. **AWT Components**: These are the building blocks of a graphical user interface (GUI) in Java. Common AWT components include buttons, text fields, labels, checkboxes, and panels, which are used to interact with the

user in a windowed application.

* + 1. **ActionListener Interface**: This is an interface in Java that listens for actions (like button clicks or menu selections) and defines the action Performed() method, which is called when an action occurs. It's used to handle events generated by user interactions with GUI

components.

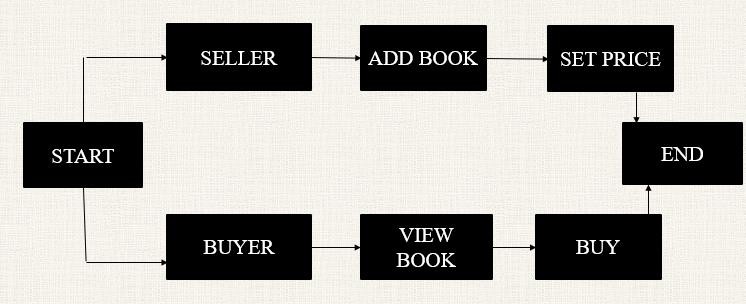
* + 1. **Event Handling**: Event handling is the process of responding to actions or events (such as a user clicking a button). In Java, event handling involves the use of event listeners (like ActionListener) to capture and respond to user actions.
    2. **Add ActionListener Method**: This method is used to attach an ActionListener to an AWT component (like a Button). It allows the component to listen for user interactions (such as a click) and trigger the action Performed**()** method when an event occurs.
    3. **Event Source and Event Listener:** In Java event handling, the event source is the component that generates an event (e.g., a Button), and the event listener is the object that handles the event (e.g., an instance of ActionListener). The listener is registered with the source using methods like add Action Listener().

# CHAPTER 2 PROJECT METHODOLOGY

## Proposed Work

Online book buying and selling in Java involves creating platforms where users can buy and sell books over the internet. Using Java for backend development ensures security, scalability, and reliability. Users can browse books, place orders, and make payments securely. Java frameworks like Spring and Hibernate are often used to streamline development. The system includes features such as user authentication, book catalogs, and transaction management. This setup allows for seamless transactions and easy management of book exchanges online

* 1. **Block Diagram**

****

# CHAPTER 3 MODULE DESCRIPTION

## User Interface (UI) Module

This module creates and organizes the visual components such as labels, buttons, and text fields using AWT. It arranges these components in a window with a Flow Layout, providing an interactive interface for the user. The UI captures user input and displays relevant information. It ensures a user-friendly experience for interacting with the application.

## Event Handling Module

The Event Handling Module listens for user actions, such as button clicks, and triggers appropriate responses. It implements the ActionListener interface, with the action Performed() method to handle events. This module updates the UI based on user interaction. It ensures the application responds dynamically to user inputs.

## Book Management Module

This module manages the collection of books using an Array List<Book> to store book objects. It defines the Book class, which includes attributes like title, author, and price. The module allows adding, viewing, and modifying book data. It handles the internal structure for managing the book collection.

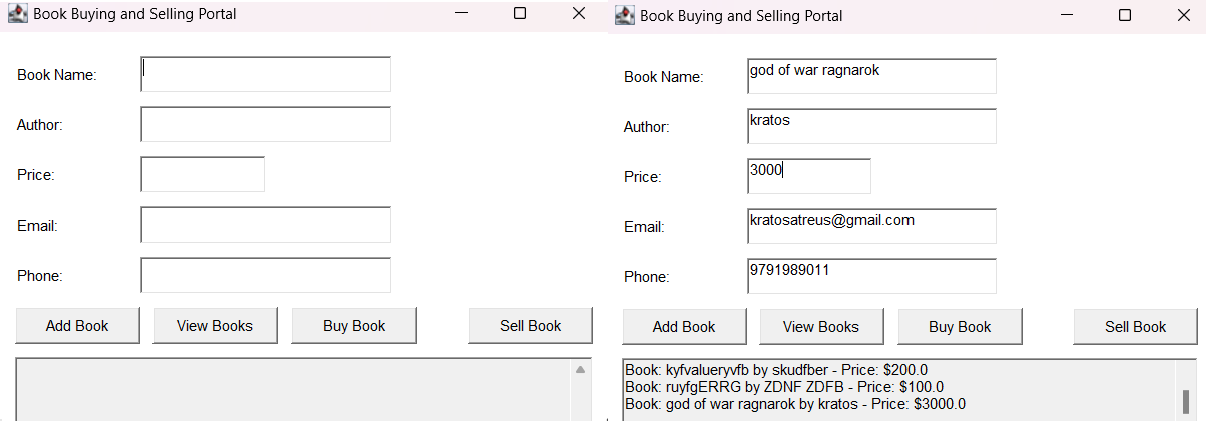
## Book Actions (Buy, Sell, Add, View) Module

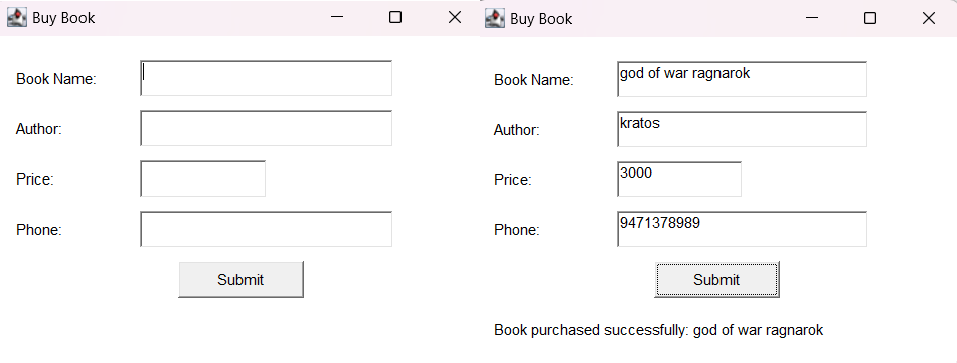
This module defines the logic for actions like adding, viewing, buying, and selling books. It processes the user's commands to update the list of books accordingly. It ensures that when a user buys or sells a book, the book list is updated correctly. This module manages the main functionality for interacting with the book data

## Window Closing Module

The Main Application Module initializes the program by creating the main window It acts as the entry point for the application, setting up the user interface and event handling. This module ensures the application is ready to run. It serves as the starting point for executing the book buying and selling system.

# CHAPTER 4 RESULTS AND DISCUSSION

****

****

**RESULTS AND DISCUSSION**

The "Online Book Portal" application effectively implements a user-friendly interface for managing the buying, selling, and viewing of books. Users can add books to the system, view available books, and purchase them with required details like book name, author, price, email, and phone. The application stores the book data in a text file, ensuring persistence across sessions. However, while the core functionality works as expected, the "Buy Book" feature could be enhanced by better tracking of the buyer’s and seller’s details and ensuring proper validation to handle cases like duplicate entries or incomplete input.

# CHAPTER 5 CONCLUSION

The project successfully demonstrates a simple online book buying and selling system using Java AWT. It provides basic functionality for adding, viewing, buying, and selling books, with a user-friendly interface and error handling. Although the system operates entirely in memory, it offers a solid foundation for further enhancement. Future improvements could include persistent storage, advanced validation, and user management features. Overall, the project serves as a functional prototype for a book management system.

# REFERENCES:

1. Java: A Beginner's Guide" by Herbert Schildt: This book is great for understanding the basics of Java, including the GUI components and event handling.
2. ”Core Java Volume I - Fundamentals" by Cay S. Horstmann: It provides detailed examples of GUI programming with Java, including file handling and event-driven programming.
3. "Java Swing" by Marc Loy: Although AWT is used in your code, Swing is its successor and is often used for building GUI applications. This book can provide insights into more advanced GUI concepts.

# APPENDIX

import java.awt.\*; import java.awt.event.\*; import java.io.\*;

import java.util.ArrayList;

public class OnlineBookPortal extends Frame implements ActionListener { private static final long serialVersionUID = 1L;

// Declare components

private TextField bookNameField, authorField, priceField, emailField, phoneField; private Button addButton, viewButton, buyButton, sellButton;

private TextArea booksArea;

private Label bookNameLabel, authorLabel, priceLabel, emailLabel, phoneLabel, statusLabel;

private ArrayList<String> booksForSale;

private static final String FILE\_NAME = "books.txt";

// Constructor to set up the GUI components public OnlineBookPortal() {

// Initialize books list

booksForSale = new ArrayList<>();

loadBooksFromFile(); // Load books from file when the app starts

// Set up frame properties

setTitle("Book Buying and Selling Portal"); setSize(500, 350);

setLayout(null); // Using absolute positioning for more control over placement

// Initialize labels

bookNameLabel = new Label("Book Name:"); authorLabel = new Label("Author:"); priceLabel = new Label("Price:");

emailLabel = new Label("Email:"); phoneLabel = new Label("Phone:"); statusLabel = new Label("");

// Initialize text fields bookNameField = new TextField(20); authorField = new TextField(20);

priceField = new TextField(10); emailField = new TextField(20); phoneField = new TextField(15);

// Initialize buttons

addButton = new Button("Add Book"); viewButton = new Button("View Books"); buyButton = new Button("Buy Book"); sellButton = new Button("Sell Book");

// Initialize TextArea for displaying books booksArea = new TextArea(10, 30); booksArea.setEditable(false);

// Set positions for components (using setBounds for absolute layout) bookNameLabel.setBounds(20, 50, 80, 30);

bookNameField.setBounds(120, 50, 200, 30);

authorLabel.setBounds(20, 90, 80, 30);

authorField.setBounds(120, 90, 200, 30);

priceLabel.setBounds(20, 130, 80, 30);

priceField.setBounds(120, 130, 100, 30);

emailLabel.setBounds(20, 170, 80, 30);

emailField.setBounds(120, 170, 200, 30);

phoneLabel.setBounds(20, 210, 80, 30);

phoneField.setBounds(120, 210, 200, 30);

right

addButton.setBounds(20, 250, 100, 30);

viewButton.setBounds(130, 250, 100, 30);

buyButton.setBounds(240, 250, 100, 30);

sellButton.setBounds(380, 250, 100, 30); // Place the "Sell Book" button on the

booksArea.setBounds(20, 290, 460, 80);

statusLabel.setBounds(20, 380, 460, 30);

// Add components to the frame add(bookNameLabel); add(bookNameField); add(authorLabel); add(authorField);

add(priceLabel); add(priceField); add(emailLabel); add(emailField); add(phoneLabel); add(phoneField); add(addButton); add(viewButton); add(buyButton); add(sellButton); add(booksArea); add(statusLabel);

// Add action listeners to buttons addButton.addActionListener(this); viewButton.addActionListener(this); buyButton.addActionListener(this); sellButton.addActionListener(this);

// Set frame visible setVisible(true);

// Close the frame when the user clicks the close button addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent we) { System.exit(0);

}

});

}

// Load books from a file

private void loadBooksFromFile() {

try (BufferedReader reader = new BufferedReader(new FileReader(FILE\_NAME))) {

String line;

while ((line = reader.readLine()) != null) { booksForSale.add(line);

}

} catch (IOException e) {

System.out.println("No previous book records found, starting fresh.");

}

}

// Save books to the file

private void saveBooksToFile() {

try (BufferedWriter writer = new BufferedWriter(new FileWriter(FILE\_NAME))) {

for (String book : booksForSale) { writer.write(book); writer.newLine();

}

} catch (IOException e) { statusLabel.setText("Error saving books to file.");

}

}

// ActionListener method to handle button clicks public void actionPerformed(ActionEvent e) {

String command = e.getActionCommand();

// Check if email and phone number fields are filled

if (emailField.getText().isEmpty() || phoneField.getText().isEmpty()) { statusLabel.setText("Email and Phone number are mandatory!"); return;

}

// Add a book to the list

if (command.equals("Add Book")) {

String bookName = bookNameField.getText(); String author = authorField.getText();

String priceText = priceField.getText();

if (bookName.isEmpty() || author.isEmpty() || priceText.isEmpty()) { statusLabel.setText("Please fill all book details.");

return;

}

price;

try {

double price = Double.parseDouble(priceText);

String book = "Book: " + bookName + " by " + author + " - Price: $" + booksForSale.add(book);

saveBooksToFile(); // Save books to file after adding statusLabel.setText("Book added successfully.");

} catch (NumberFormatException ex) { statusLabel.setText("Invalid price entered.");

}

// Clear text fields after adding bookNameField.setText(""); authorField.setText(""); priceField.setText("");

}

// View the available books for sale

else if (command.equals("View Books")) { if (booksForSale.isEmpty()) {

booksArea.setText("No books available for sale.");

} else {

booksArea.setText("");

for (String book : booksForSale) { booksArea.append(book + "\n");

}

}

statusLabel.setText("Viewing books for sale.");

}

// Handle buying a book

else if (command.equals("Buy Book")) {

new BuyBookWindow(); // Open the new frame for buying a book

}

// Handle selling a book

else if (command.equals("Sell Book")) {

String bookName = bookNameField.getText(); String author = authorField.getText();

String priceText = priceField.getText();

if (bookName.isEmpty() || author.isEmpty() || priceText.isEmpty()) { statusLabel.setText("Please fill all fields.");

return;

}

try {

double price = Double.parseDouble(priceText);

String book = "Book: " + bookName + " by " + author + " - Price: $" + price;

booksForSale.add(book);

saveBooksToFile(); // Save books after selling statusLabel.setText("Successfully listed the book for sale: " + bookName);

} catch (NumberFormatException ex) { statusLabel.setText("Invalid price entered.");

}

// Clear text fields after selling bookNameField.setText(""); authorField.setText(""); priceField.setText("");

}

}

// Inner class to create the "Buy Book" window class BuyBookWindow extends Frame {

private static final long serialVersionUID = 1L;

private TextField bookNameField, authorField, priceField, phoneField; private Button submitButton;

private Label bookNameLabel, authorLabel, priceLabel, phoneLabel, statusLabel;

public BuyBookWindow() {

// Set up frame properties setTitle("Buy Book"); setSize(400, 300);

setLayout(null); // Use null layout for absolute positioning

// Initialize labels and fields

bookNameLabel = new Label("Book Name:"); authorLabel = new Label("Author:"); priceLabel = new Label("Price:");

phoneLabel = new Label("Phone:"); statusLabel = new Label("");

bookNameField = new TextField(20); authorField = new TextField(20); priceField = new TextField(10); phoneField = new TextField(15);

submitButton = new Button("Submit");

// Set positions for components (using setBounds for absolute layout) bookNameLabel.setBounds(20, 50, 80, 30);

bookNameField.setBounds(120, 50, 200, 30);

authorLabel.setBounds(20, 90, 80, 30);

authorField.setBounds(120, 90, 200, 30);

priceLabel.setBounds(20, 130, 80, 30);

priceField.setBounds(120, 130, 100, 30);

phoneLabel.setBounds(20, 170, 80, 30);

phoneField.setBounds(120, 170, 200, 30);

submitButton.setBounds(150, 210, 100, 30);

statusLabel.setBounds(20, 250, 360, 30);

// Add components to the frame add(bookNameLabel); add(bookNameField); add(authorLabel); add(authorField); add(priceLabel); add(priceField); add(phoneLabel); add(phoneField); add(submitButton); add(statusLabel);

// Add action listener to submit button submitButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) { String bookName = bookNameField.getText(); String author = authorField.getText();

String priceText = priceField.getText(); String phone = phoneField.getText();

// Check if all fields are filled

if (bookName.isEmpty() || author.isEmpty() || priceText.isEmpty() || phone.isEmpty()) {

statusLabel.setText("Please fill all the details."); return;

}

try {

double price = Double.parseDouble(priceText); statusLabel.setText("Book purchased successfully: " + bookName);

} catch (NumberFormatException ex) { statusLabel.setText("Invalid price entered.");

}

}

});

// Set frame visible setVisible(true);

}

}

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

}

}