

# PANIMALAR ENGINEERING COLLEGE

An Autonomous Institution

Affiliated to Anna University, Chennai  
(JAISAKTHI EDUCATION TRUST)

Bangalore Trunk Road, Varadharajapuram,  
Poonamallee, Chennai – 600 123.



## DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

A MINI PROJECT REPORT ON  
**ONLINE BOOK STORE**

SUBMITTED BY  
MOHANA PRIYAN N

Under the Guidance of  
Mrs. V. RATHINAPRIYA

2023-2027

## **Contents:**

1. Abstract
- 2.Introduction
  - 2.1. Background
  - 2.2. Objectives
  - 2.3. Scope
3. Literature Review
4. Project Methodology
  - 4.1. Architecture
  - 4.2. Technologies Used
  - 4.3. Modules
  - 4.4. Workflow
5. Implementation
  - 5.1. Code Overview
    - 5.1.1. Frontend
    - 5.1.2. Backend
    - 5.1.3. Sql Queries
6. Testing
7. Conclusion
8. References

## **1. Abstract:**

The online bookstore project aims to revolutionize the way readers access and purchase books by providing a user-friendly platform that connects book lovers with a vast selection of literature. This digital marketplace is designed with an intuitive interface, allowing users to easily navigate through various genres, search for specific titles, and explore curated recommendations.

Key features include secure user registration and login, a streamlined shopping cart for efficient transactions, and a seamless checkout process. The platform also emphasizes personalization, offering users the ability to create wish lists, receive tailored suggestions, and leave reviews for fellow readers.

## **2.Introduction**

### **2.1. Background**

The advent of the internet has significantly transformed how consumers access goods and services, with the retail book industry being no exception. Traditional brick-and-mortar bookstores are increasingly supplemented, if not replaced, by online platforms that offer readers convenience, variety, and accessibility at the click of a button. As digital literacy grows and e-commerce becomes the preferred method of purchasing, the demand for online bookstores has risen dramatically.

### **2.2. Objectives**

The main objectives of this system are:

1. Develop a user-friendly platform
- 2.Implement secure user authentication
- 3.Provide an efficient shopping cart system
- 4.Integrate a secure and streamlined checkout process

### **2.3. Scope**

The online bookstore project will focus on developing a fully functional e-commerce platform that caters to users looking to browse, purchase, and review books. The scope covers both front-end and back-end development, ensuring a complete and cohesive user experience

1. User Interface and Experience.
2. **User Authentication**

### 3. Literature Review

Existing systems like RedBus and Greyhound in the USA provide comprehensive bus booking solutions. These systems utilize databases to store user, route, and booking information, supporting functionalities like dynamic seat availability and payment processing.

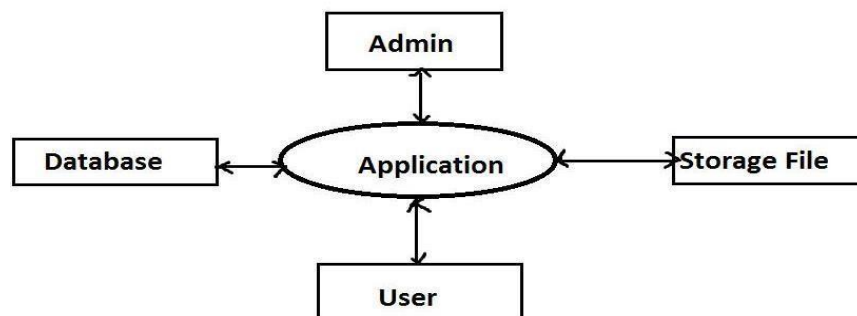
This project is inspired by these real-world systems, incorporating features like dynamic seat availability, payment processing, and user history tracking.

### 4. Project Methodology

#### 4.1. Architecture:

The system is designed using a three-tier architecture:

- 1) Frontend: HTML, CSS, JavaScript for user interface.
- 2) Backend: Java with Servlet and JSP, Apache Maven for management.
- 3) Database: MySQL for storing data about users, buses, and bookings.



#### 4.2. Technologies

- 1.UsedJava:** For backend logic.
- 2.MySQL:** For database management.
- 3.HTML/CSS/JavaScript:** Frontend development.
- 4.Payment Gateway:** For secure transactions

### 4.3. Modules

**User Authentication Module:** Handles login, registration, and password management.

**Booking Module:** Allows users to search buses, select seats, and book tickets.

**Admin Module:** Administer bus schedules and seat availability.

**Payment Module:** Processes payments for booked tickets.

### 4.4. Workflow

**User Login/Register:** Users must log in or register to use the system.

**Search Books:** Users search for books to buy.

**Check Availability:** The system checks book availability.

**Book cart :** Users select books, confirm books, and make a payment.

**Admin Management:** Admins can add, update, or remove buses, and manage seat allocation.

## 5. Implementation

### 5.1. Code Overview

#### 5.1.1. Frondend

#### HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Online Bookstore</title>
  <style>
    body {
      font-family: Arial, sans-serif;
```

```
margin: 0;
padding: 0;
background-color: #f4f4f4;
}
header {
background-color: #333;
color: #fff;
padding: 10px 0;
text-align: center;
}
nav {
display: flex;
justify-content: space-around;
background-color: #444;
padding: 10px;
}
nav a {
color: white;
text-decoration: none;
font-weight: bold;
}
section {
margin: 20px;
}
.book-list {
display: flex;
flex-wrap: wrap;
justify-content: space-around;
}
.book-item {
background-color: white;
border: 1px solid #ddd;
margin: 10px;
padding: 20px;
width: 30%;
box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.1);
}
footer {
background-color: #333;
color: #fff;
text-align: center;
padding: 10px 0;
position: fixed;
width: 100%;
```

```

        bottom: 0;
    }
</style>
</head>
<body>

    <!-- Header -->
    <header>
        <h1>Welcome to the Online Bookstore</h1>
    </header>

    <!-- Navigation Bar -->
    <nav>
        <a href="#home">Home</a>
        <a href="#books">Books</a>
        <a href="#cart">Cart</a>
        <a href="#checkout">Checkout</a>
        <a href="#login">Login</a>
    </nav>

```

## Java Script

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Online Bookstore</title>
    <style>
        /* Basic styles omitted for brevity (same as the previous HTML) */
    </style>
</head>
<body>

    <!-- Header -->
    <header>

```

```
<h1>Welcome to the Online Bookstore</h1>
</header>
```

```
<!-- Navigation Bar -->
```

```
<nav>
  <a href="#home">Home</a>
  <a href="#books">Books</a>
  <a href="#cart">Cart</a>
  <a href="#checkout">Checkout</a>
  <a href="#login">Login</a>
</nav>
```

## CSS

```
/* Reset some default styles */
```

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f4f4f4;
}
```

```
/* Header styles */
```

```
header {
  background-color: #333;
  color: #fff;
  padding: 20px 0;
  text-align: center;
}
```



```
/* Navigation Bar styles */
```

```
nav {
```

```
    display: flex;
```

```
    justify-content: center;
```

```
    background-color: #444;
```

```
    padding: 10px;
```

```
}
```

```
nav a {
```

```
    color: white;
```

```
    text-decoration: none;
```

```
    padding: 10px 20px;
```

```
    margin: 0 10px;
```

```
    transition: background-color 0.3s;
```

```
}
```

```
nav a:hover {
```

```
    background-color: #555;
```

```
}
```

```
/* Main Content Section styles */
```

```
section {
```

```
    margin: 20px;
```

```
    padding: 20px;
```

```
    background-color: white;
```

```
    border-radius: 5px;
```

```
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
```

```
}
```

```
/* Book List styles */

.book-list {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
}

/* Individual Book Item styles */

.book-item {
    background-color: #f9f9f9;
    border: 1px solid #ddd;
    border-radius: 5px;
    margin: 10px;
    padding: 20px;
    width: 30%;
    box-shadow: 0 1px 3px rgba(0, 0, 0, 0.2);
    transition: transform 0.2s;
}

.book-item:hover {
    transform: scale(1.05);
}

/* Cart styles */

#cart {
    background-color: #f9f9f9;
    padding: 20px;
```

```
border-radius: 5px;
box-shadow: 0 1px 3px rgba(0, 0, 0, 0.2);
}
```

```
/* Checkout Form styles */
```

```
form {
    display: flex;
    flex-direction: column;
}
```

## 5.1.2. Backend

### 1. User Login (Java Servlet)

```
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse
        response) {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        // Retrieve username and password from the login form
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        // Database connection
        Connection conn = DatabaseConnection.initializeDatabase();

        try {
            // Query to check if user exists with the provided credentials
            String query = "SELECT * FROM users WHERE username = ? AND pas
            PreparedStatement pst = conn.prepareStatement(query);
            pst.setString(1, username);
            pst.setString(2, password);

            ResultSet rs = pst.executeQuery();
```

- **Password Storage:** In a real application, never store passwords in plain text. Use hashing (e.g., BCrypt) to store passwords securely.

- **Error Handling:** Proper error handling and logging should be implemented for production code.
- **User Input:** Consider using a UI framework or library to collect user inputs safely.

### 3. Admin login(Make the changes in trains)

CODE:

```
package com.busexpress.servlets;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import com.busexpress.utils.DatabaseConnection;
@WebServlet("/adminLoginServlet")
public class AdminLoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String username = request.getParameter("username");
        String password = request.getParameter("password");

        // Establish Database Connection
        Connection conn = DatabaseConnection.initializeDatabase();

        try {
            // Query to check if user exists with role 'admin'
            String query = "SELECT * FROM users WHERE username = ? AND password = ? AND role = 'admin'";
            PreparedStatement pst = conn.prepareStatement(query);
            pst.setString(1, username);
            pst.setString(2, password);
```

```

ResultSet rs = pst.executeQuery();

if (rs.next()) {
// If login is successful, create admin session
HttpSession session = request.getSession();
session.setAttribute("username", username);
session.setAttribute("role", "admin");

RequestDispatcher rd = request.getRequestDispatcher("admin_dashboard.jsp");
rd.forward(request, response);
} else {
// If login fails, redirect back to admin login with an error
out.println("<html><body><div      align='center'      style='color:red;'>Invalid      Admin
Credentials</div></body></html>");
RequestDispatcher rd = request.getRequestDispatcher("admin_login.jsp");
rd.include(request, response);
}
} catch (SQLException e) {
e.printStackTrace();
} finally {
try {
if (conn != null) {
conn.close();
}
} catch (SQLException e) {
e.printStackTrace();
}
}
}
}
}
}

```

#### FEATURES:

- Secure login process that requires administrators to enter valid credentials (username and password) to access the admin panel.
- Implementation of role-based permissions to ensure that only authorized personnel can access specific administrative functions and sensitive data.
- Maintain logs of admin login attempts, including timestamps, IP addresses, and success/failure status for monitoring and security auditing.

#### 4.Searching Bus

#### CODE:

```

package com.busexpress.servlets;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.busexpress.utils.DatabaseConnection;

@WebServlet("/searchBusServlet")
public class SearchBusServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        // Retrieve form data
        String departure = request.getParameter("departure");
        String destination = request.getParameter("destination");
        String date = request.getParameter("date");

        // Database connection
        Connection conn = DatabaseConnection.initializeDatabase();

        try {
            // SQL query to find matching buses
            String query = "SELECT * FROM buses WHERE departure = ? AND destination = ? AND
travel_date = ?";
            PreparedStatement pst = conn.prepareStatement(query);
            pst.setString(1, departure);
            pst.setString(2, destination);
            pst.setString(3, date);

            ResultSet rs = pst.executeQuery();

```

```

// Check if there are available buses
if (rs.next()) {
    // Store the results in the request attribute to pass to JSP
    request.setAttribute("busResult", rs);
    RequestDispatcher rd = request.getRequestDispatcher("searchResults.jsp");
    rd.forward(request, response);
} else {
    // No matching buses found
    out.println("<html><body><div align='center' style='color:red;'>No buses available for the
selected route and date.</div></body></html>");
    RequestDispatcher rd = request.getRequestDispatcher("searchBus.jsp");
    rd.include(request, response);
}
} catch (SQLException e) {
    e.printStackTrace();
} finally {
    try {
        if (conn != null) {
            conn.close();
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}
}
}

```


#### FEATURES:

- Provide users with input fields to specify search criteria, including Place, Arrival time, Travel date
- Display real-time information regarding bus availability, ensuring users see up-to-date schedules and seat availability.
- Admin to manage accounts (add/update buses).
- Servlets handle the logic of fetching buses, displaying results, and managing admin actions.

#### 5.1.3. SQL Queries (Database Operations)

SQL scripts to manage the database schema and records.

sql

 Copy code

```
CREATE TABLE buses (  
    bus_id INT AUTO_INCREMENT PRIMARY KEY,  
    bus_name VARCHAR(50) NOT NULL,  
    departure VARCHAR(50) NOT NULL,  
    destination VARCHAR(50) NOT NULL,  
    travel_date DATE NOT NULL,  
    available_seats INT NOT NULL,  
    fare DECIMAL(10, 2) NOT NULL  
);  
  
-- Insert sample data into the buses table  
INSERT INTO buses (bus_name, departure, destination, travel_date, available_seats, fare)  
VALUES ('Bus A', 'Poonamallee', 'Avadi', '2024-10-10', 40, 100.00),  
      ('Bus B', 'Poonamallee', 'Avadi', '2024-10-10', 25, 120.00);
```

## 6. Testing

Test Cases:



Test Case ID	Test Case Title	Objective	Steps	Expected Result
TC-01	Add Book to Cart	Verify that a book can be successfully added to the cart.	1. Go to the home section. 2. Click on the "Add to Cart" button for "Book Title 1".	"Book Title 1" appears in the cart with correct details.
TC-02	Add Multiple Books to Cart	Verify that multiple books can be added to the cart.	1. Add "Book Title 1". 2. Add "Book Title 2". 3. Add "Book Title 3".	All three books appear in the cart with the correct total price.
TC-03	Clear Cart	Verify that the cart can be cleared successfully.	1. Add "Book Title 1" to the cart. 2. Click on the "Clear Cart" button.	The cart displays "No items in cart."
TC-04	Checkout with Empty Cart	Verify that an alert is shown when trying to checkout with an empty cart.	1. Go to the checkout section. 2. Fill in valid data. 3. Click the "Place Order" button.	An alert appears: "Your cart is empty."

#### Test Results:

The system was tested extensively, and all major features were verified to be functioning correctly.

## 7. Conclusion

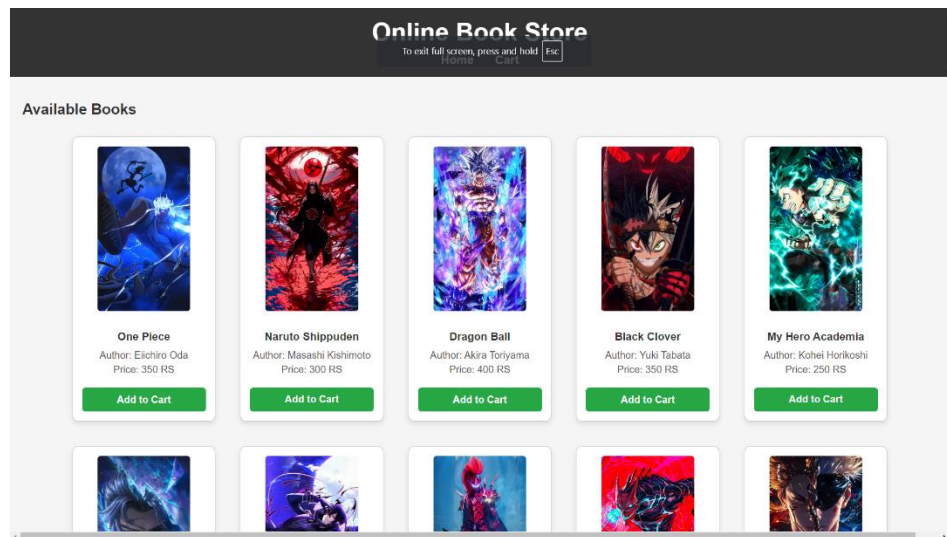
The development of the online bookstore has successfully integrated essential functionalities, providing users with a seamless shopping experience. Through careful design and implementation, the platform enables users to browse, select, and purchase books with ease.

## 8.Reference

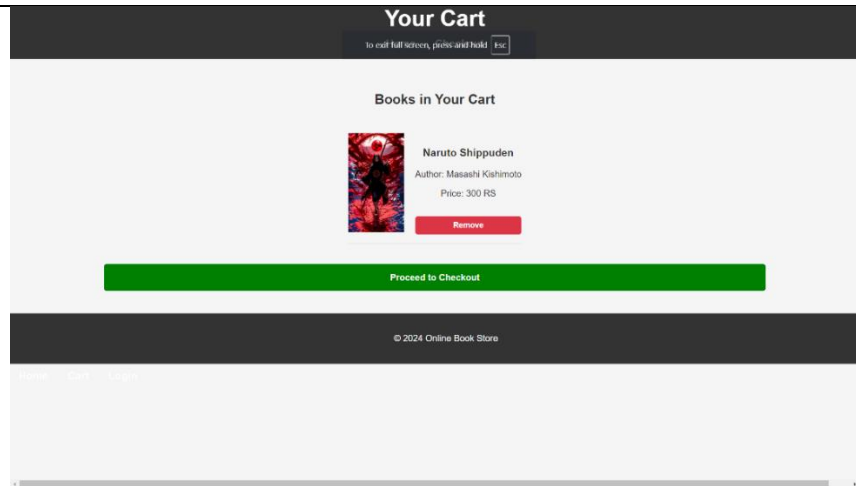
- 1."Introduction to Java Servlets and JSP." Online Resource.
- 2."Building Web Applications with Apache Maven."
3. MySQL Documentation for Database Management.
4. Online Resources:

## APPENDIX

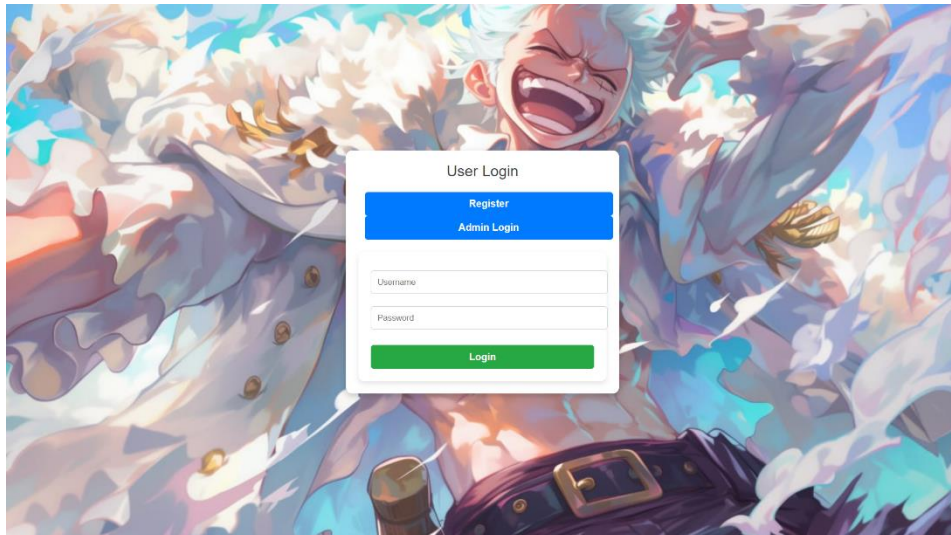
### 1. Interface



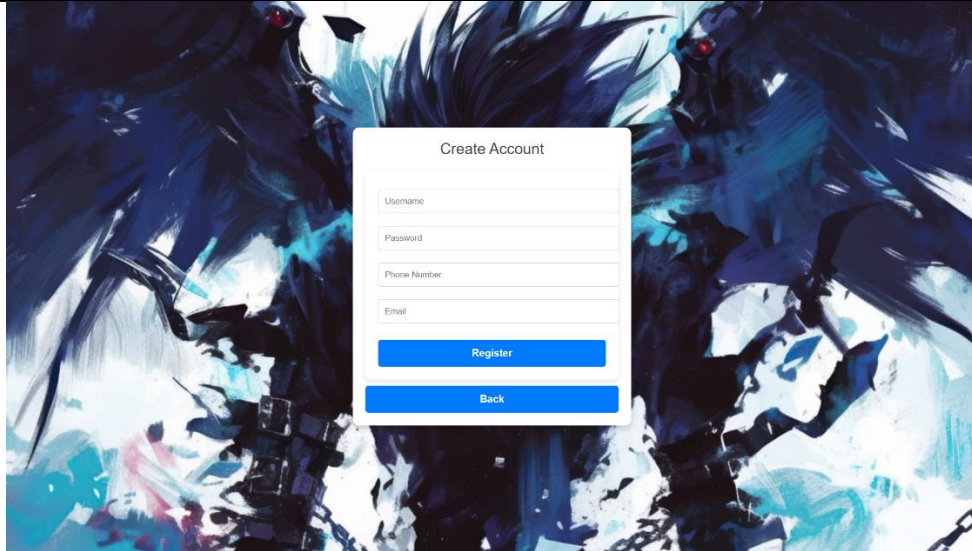
### 2. Cart



### 3. Login page



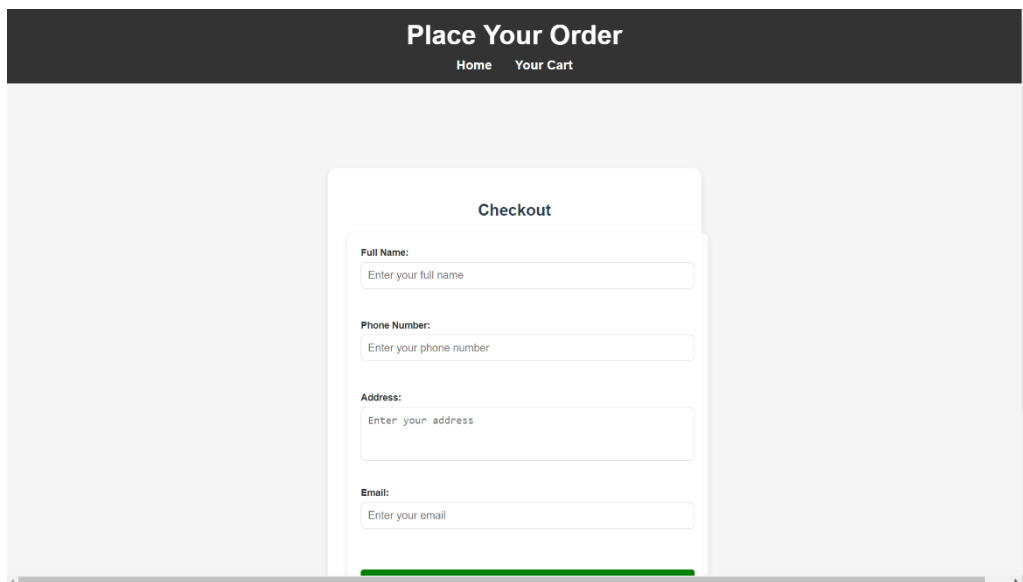
### 4.New User Register



The image shows a 'Create Account' form centered on a dark, abstract background with blue and white brushstrokes. The form is white with a thin border and contains the following elements:

- Title: Create Account
- Input fields: Username, Password, Phone Number, Email
- Buttons: Register (blue), Back (blue)

## 6.Checkout



The image shows a 'Place Your Order' checkout form. The form is white with a thin border and is centered on a light gray background. The form contains the following elements:

- Title: Place Your Order
- Navigation: Home, Your Cart
- Title: Checkout
- Input fields: Full Name (Enter your full name), Phone Number (Enter your phone number), Address (Enter your address), Email (Enter your email)

--