**Web Programming - Exercise 1**

**Aim:**

To Design a web site using HTML and DHTML. Use Internal hyperlinking, basic text formatting, images, forms, frames, links, tables, CSS, animations.

**Required web tools and methodology:**

Frontend Development:

● HTML: For structuring the content and creating forms, tables, and navigation menus.

● CSS: For styling and layout of the web pages.

● JavaScript (DHTML): For dynamic features like animations, form validations, and interactivity.

**Implementation procedure**

1. Install a text editor (e.g., Visual Studio Code) and set up a local server (e.g., XAMPP, Flask, or Django).
2. Create a home page using HTML to serve as the central hub with internal links to other sections.
3. Design pages for Product Details, Purchase Records, Sales Records, and Stock Overview using HTML and CSS.
4. Implement navigation menus with internal hyperlinks for seamless page transitions.
5. Use CSS to style the website, including tables, forms, and navigation bars.
6. Create tables in HTML to display Product, Purchase, Sales, and Stock details.
7. Add forms in HTML for entering product, purchase, and sales data.
8. Use JavaScript to validate form inputs and enhance user interactivity.
9. Write JavaScript for DHTML animations like dynamic menu highlights and hover effects.
10. Add animations using JavaScript or CSS for visual appeal.
11. Test the entire application to ensure all links, forms, and dynamic elements work as expected.
12. Deploy the website

**Code:**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>Web Stock Maintenance</title>**

**<style>**

**body {**

**font-family: Arial, sans-serif;**

**margin: 0;**

**padding: 0;**

**background-color: #f9f9f9;**

**}**

**header {**

**background-color: #0E4AA0;**

**color: white;**

**padding: 10px;**

**text-align: center;**

**}**

**nav {**

**margin-top: 10px;**

**}**

**nav a {**

**color: white;**

**text-decoration: none;**

**margin: 0 10px;**

**padding: 5px 10px;**

**background-color: #0056b3;**

**border-radius: 5px;**

**}**

**nav a:hover {**

**background-color: #003f88;**

**}**

**section {**

**margin: 20px;**

**padding: 20px;**

**background-color: #ffffff;**

**border: 1px solid #ddd;**

**border-radius: 5px;**

**}**

**form {**

**margin-bottom: 20px;**

**}**

**form label {**

**display: block;**

**margin-bottom: 10px;**

**}**

**button {**

**background-color: #0E4AA0;**

**color: white;**

**border: none;**

**padding: 10px 20px;**

**border-radius: 5px;**

**cursor: pointer;**

**}**

**button:hover {**

**background-color: #0056b3;**

**}**

**table {**

**width: 100%;**

**border-collapse: collapse;**

**margin-top: 20px;**

**}**

**table th,**

**table td {**

**border: 1px solid #ddd;**

**padding: 10px;**

**text-align: left;**

**}**

**table th {**

**background-color: #f4f4f4;**

**}**

**</style>**

**</head>**

**<body>**

**<header>**

**<h1>Web Stock Maintenance System</h1>**

**<nav>**

**<a href="#product">Product Details</a>**

**<a href="#purchase">Purchase Details</a>**

**<a href="#sales">Sales Details</a>**

**<a href="#stock">Stock Details</a>**

**</nav>**

**</header>**

**<main>**

**<section id="product">**

**<h2>Product Details</h2>**

**<form>**

**<label>Product Code: <input type="text" name="product\_code"></label>**

**<label>Product Name: <input type="text" name="product\_name"></label>**

**<label>Opening Stock: <input type="number" name="opening\_stock"></label>**

**<label>Price: <input type="number" name="price" step="0.01"></label>**

**<button type="submit">Add Product</button>**

**</form>**

**<table>**

**<tr>**

**<th>Product Code</th>**

**<th>Product Name</th>**

**<th>Opening Stock</th>**

**<th>Price</th>**

**</tr>**

**<!-- Rows dynamically added -->**

**</table>**

**</section>**

**<section id="purchase">**

**<h2>Purchase Details</h2>**

**<form>**

**<label>Product Code: <input type="text" name="product\_code"></label>**

**<label>Quantity: <input type="number" name="quantity"></label>**

**<label>Price: <input type="number" name="price" step="0.01"></label>**

**<button type="submit">Log Purchase</button>**

**</form>**

**</section>**

**<section id="sales">**

**<h2>Sales Details</h2>**

**<form>**

**<label>Date: <input type="date" name="date"></label>**

**<label>Customer Name: <input type="text" name="customer\_name"></label>**

**<label>Product Code: <input type="text" name="product\_code"></label>**

**<label>Quantity: <input type="number" name="quantity"></label>**

**<label>Price: <input type="number" name="price" step="0.01"></label>**

**<button type="submit">Log Sale</button>**

**</form>**

**</section>**

**<section id="stock">**

**<h2>Stock Details</h2>**

**<table>**

**<tr>**

**<th>Product ID</th>**

**<th>Opening Stock</th>**

**<th>Purchase Stock</th>**

**<th>Sales Stock</th>**

**<th>Current Stock</th>**

**</tr>**

**<!-- Rows dynamically added -->**

**</table>**

**</section>**

**</main>**

**<footer>**

**<p>&copy; 2025 Web Stock Maintenance</p>**

**</footer>**

**<script>**

**document.addEventListener("DOMContentLoaded", () => {**

**const buttons = document.querySelectorAll("button");**

**buttons.forEach(button => {**

**button.addEventListener("mouseover", () => {**

**button.style.transform = "scale(1.1)";**

**});**

**button.addEventListener("mouseout", () => {**

**button.style.transform = "scale(1)";**

**});**

**});**

**});**

**document.addEventListener("DOMContentLoaded", () => {**

**const productForm = document.getElementById("productForm");**

**const productTable = document.getElementById("productTable").querySelector("tbody");**

**const stockTable = document.getElementById("stockTable").querySelector("tbody");**

**const stockData = {}; // Object to store stock details (keyed by Product Code)**

**// Handle Product Form Submission**

**productForm.addEventListener("submit", (event) => {**

**event.preventDefault(); // Prevent form refresh**

**// Retrieve input values**

**const productCode = document.getElementById("productCode").value.trim();**

**const productName = document.getElementById("productName").value.trim();**

**const openingStock = parseInt(document.getElementById("openingStock").value.trim());**

**const price = parseFloat(document.getElementById("price").value.trim());**

**// Create a new row in the Product table**

**const newRow = document.createElement("tr");**

**newRow.innerHTML = `**

**<td>${productCode}</td>**

**<td>${productName}</td>**

**<td>${openingStock}</td>**

**<td>${price}</td>**

**`;**

**productTable.appendChild(newRow);**

**// Add/Update Stock Details**

**if (!stockData[productCode]) {**

**stockData[productCode] = {**

**openingStock: openingStock,**

**purchaseStock: 0,**

**salesStock: 0,**

**currentStock: openingStock,**

**};**

**// Add row to Stock Details table**

**const stockRow = document.createElement("tr");**

**stockRow.setAttribute("data-product-code", productCode);**

**stockRow.innerHTML = `**

**<td>${productCode}</td>**

**<td>${openingStock}</td>**

**<td>0</td>**

**<td>0</td>**

**<td>${openingStock}</td>**

**`;**

**stockTable.appendChild(stockRow);**

**}**

**// Clear the form inputs**

**productForm.reset();**

**});**

**// Function to update stock details**

**function updateStockTable(productCode, purchaseQuantity = 0, salesQuantity = 0) {**

**const stockRow = stockTable.querySelector(`tr[data-product-code="${productCode}"]`);**

**if (stockRow) {**

**const stock = stockData[productCode];**

**// Update stock values**

**stock.purchaseStock += purchaseQuantity;**

**stock.salesStock += salesQuantity;**

**stock.currentStock = stock.openingStock + stock.purchaseStock - stock.salesStock;**

**// Update table row**

**stockRow.children[2].textContent = stock.purchaseStock;**

**stockRow.children[3].textContent = stock.salesStock;**

**stockRow.children[4].textContent = stock.currentStock;**

**}**

**}**

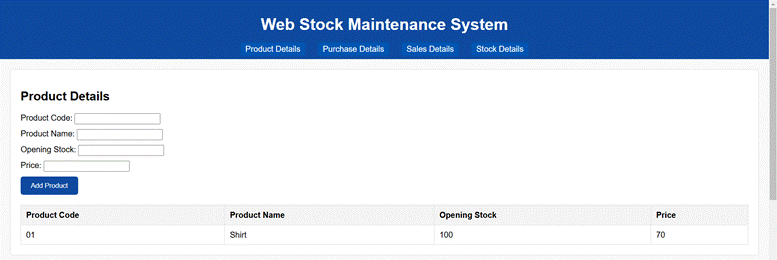
**});**

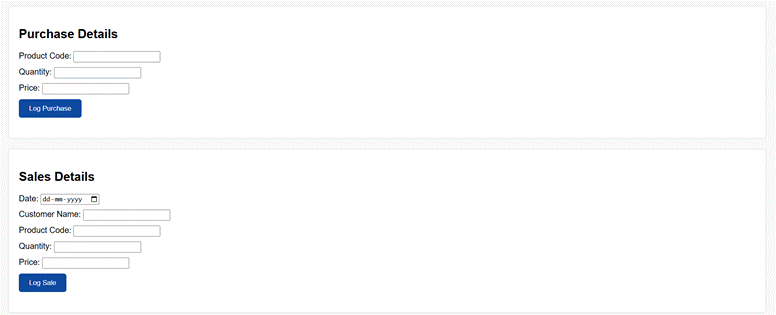
**</script>**

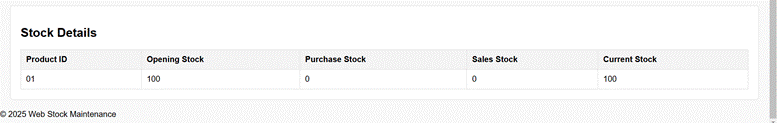
**</body>**

**</html>**

**Output:**

****

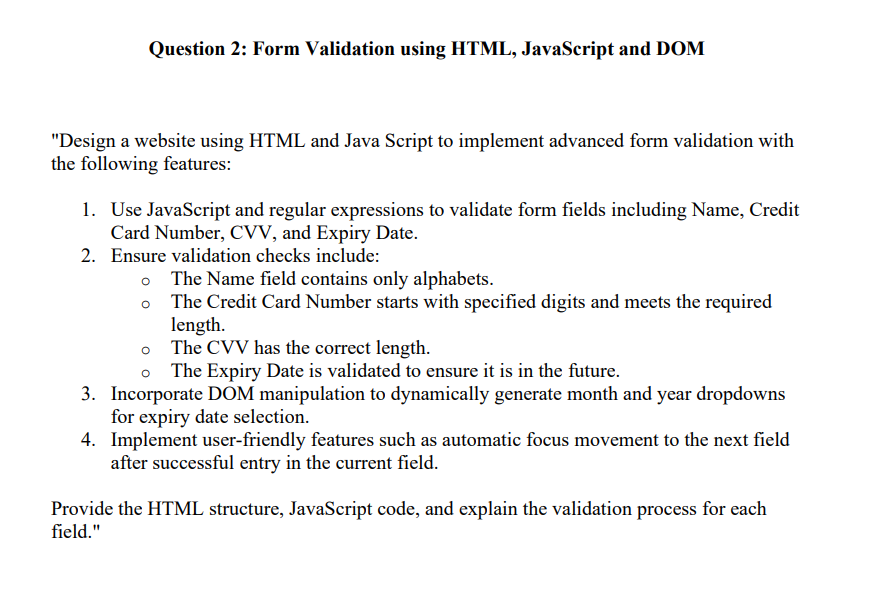
****

****

**Conclusion**

Thus, a dynamic web-based Stock Maintenance System that efficiently manages product, purchase, sales, and stock details using HTML, DHTML, CSS, and JavaScript was successfully developed. The system ensures real-time updates and a user-friendly interface, providing a robust solution for inventory management.

**Web Programming - Exercise 2**

****

**Implementation:**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>Forms</title>**

**<style>**

**body {**

**display: flex;**

**justify-content: center;**

**align-items: center;**

**height: 100vh;**

**margin: 0;**

**background-color: #f4f4f9;**

**}**

**.form-elements {**

**background-color: #fff;**

**padding: 20px;**

**border-radius: 10px;**

**box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);**

**max-width: 300px;**

**width: 100%;**

**}**

**label,**

**input {**

**display: block;**

**margin-bottom: 8px;**

**width: 200px;**

**}**

**select {**

**width: 200px;**

**display: block;**

**margin-bottom: 8px;**

**}**

**button {**

**width: 100%;**

**padding: 10px;**

**border-radius: 5px;**

**cursor: pointer;**

**}**

**</style>**

**</head>**

**<body>**

**<div class="form-elements">**

**<H2>Form Validation Example</H2>**

**<form id="form">**

**<label for="name">Name: </label>**

**<input type="text" id="name" placeholder="Enter your name">**

**<div id="name-error" class="error"></div>**

**<label for="credit">Credit Card Number: </label>**

**<input type="text" id="credit" placeholder="Enter your credit card number" maxlength="16">**

**<div id="credit-error" class="error"></div>**

**<label for="cvv">CVV: </label>**

**<input type="text" id="cvv" placeholder="Enter CVV" maxlength="3">**

**<div id="cvv-error" class="error"></div>**

**<label for="expiry">Expiry Date:</label>**

**<select id="month"></select>**

**<select id="year"></select>**

**<div id="expiry-error" class="error"></div>**

**<button type="submit">Submit</button>**

**</form>**

**</div>**

**<script>**

**function populateDropdowns() {**

**const monthDropdown = document.getElementById("month");**

**const yearDropdown = document.getElementById("year");**

**const currentYear = new Date().getFullYear();**

**for (let i = 1; i <= 12; i++) {**

**const option = document.createElement("option");**

**option.value = i;**

**option.textContent = i;**

**monthDropdown.appendChild(option);**

**}**

**for (let year = currentYear; year <= currentYear + 20; year++) {**

**const option = document.createElement("option");**

**option.value = year;**

**option.textContent = year;**

**yearDropdown.appendChild(option);**

**}**

**}**

**populateDropdowns();**

**const form = document.getElementById("form");**

**const nameInput = document.getElementById("name");**

**const creditInput = document.getElementById("credit");**

**const cvvInput = document.getElementById("cvv");**

**const monthDropdown = document.getElementById("month");**

**const yearDropdown = document.getElementById("year");**

**const nameError = document.getElementById("name-error");**

**const creditError = document.getElementById("credit-error");**

**const cvvError = document.getElementById("cvv-error");**

**const expiryError = document.getElementById("expiry-error");**

**function validateName() {**

**const nameRegex = /^[A-Za-z\s]+$/;**

**if (!nameRegex.test(nameInput.value)) {**

**nameError.textContent = "Name must contain only alphabets.";**

**return false;**

**}**

**nameError.textContent = "";**

**return true;**

**}**

**function validateCreditCard() {**

**const creditRegex = /^4[0-9]{15}|5[1-5][0-9]{14}$/; // Visa (4) or MasterCard (5)**

**if (!creditRegex.test(creditInput.value)) {**

**creditError.textContent = "Invalid Credit Card Number.";**

**return false;**

**}**

**creditError.textContent = "";**

**return true;**

**}**

**function validateCVV() {**

**const cvvRegex = /^[0-9]{3}$/;**

**if (!cvvRegex.test(cvvInput.value)) {**

**cvvError.textContent = "CVV must be 3 digits.";**

**return false;**

**}**

**cvvError.textContent = "";**

**return true;**

**}**

**function validateExpiryDate() {**

**const currentMonth = new Date().getMonth() + 1;**

**const currentYear = new Date().getFullYear();**

**const selectedMonth = parseInt(monthDropdown.value, 10);**

**const selectedYear = parseInt(yearDropdown.value, 10);**

**if (selectedYear <= currentYear || (selectedMonth <= currentMonth)) {**

**expiryError.textContent = "Expiry date must be in the future.";**

**return false;**

**}**

**expiryError.textContent = "";**

**return true;**

**}**

**form.addEventListener("submit", (event) => {**

**event.preventDefault();**

**const isNameValid = validateName();**

**const isCreditValid = validateCreditCard();**

**const isCVVValid = validateCVV();**

**const isExpiryValid = validateExpiryDate();**

**if (isNameValid && isCreditValid && isCVVValid && isExpiryValid) {**

**alert("Form submitted successfully!");**

**} else {**

**alert("Please correct the errors in the form.");**

**}**

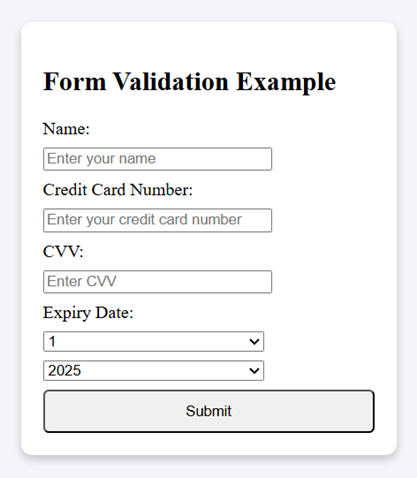
**});**

**</script>**

**</body>**

**</html>**

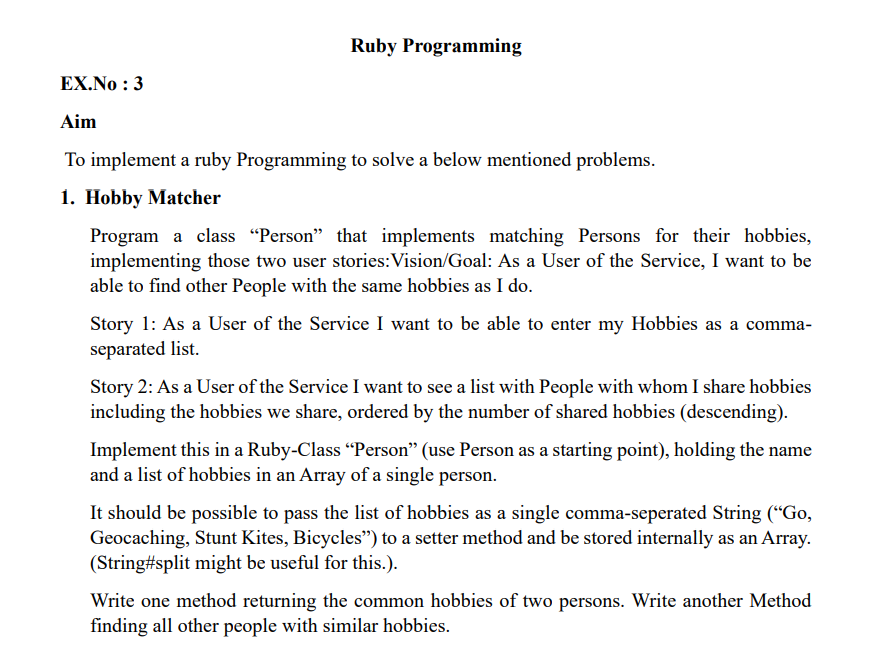
**Output**

****

**Conclusion**

Thus, a responsive and visually appealing portfolio website was successfully developed using HTML, CSS, and JavaScript. The website effectively showcases projects, skills, and contact details, offering a user-friendly interface and seamless navigation across devices.

**Web Programming - Exercise 3**

****

**Implementation:**

class Person

attr\_reader :name, :hobbies

# Initialize with a name and hobbies (which can be a comma-separated string)

def initialize(name)

@name = name

@hobbies = []

end

# Setter method to accept hobbies as a comma-separated string

def set\_hobbies(hobbies\_string)

@hobbies = hobbies\_string.split(',').map(&:strip)

end

# Method to find common hobbies with another person

def common\_hobbies(other\_person)

@hobbies & other\_person.hobbies

end

# Method to find all other people with similar hobbies

def find\_people\_with\_common\_hobbies(people)

people.reject { |person| person == self }

.map { |person| { person.name => self.common\_hobbies(person) } }

.sort\_by { |shared\_hobbies| -shared\_hobbies.values.flatten.count }

end

# Display the hobbies for a person (useful for debugging)

def display\_hobbies

puts "#{@name}'s hobbies: #{@hobbies.join(', ')}"

end

end

# Sample usage

person1 = Person.new("Alice")

person1.set\_hobbies("Go, Geocaching, Stunt Kites, Bicycles")

person2 = Person.new("Bob")

person2.set\_hobbies("Bicycles, Geocaching, Hiking")

person3 = Person.new("Charlie")

person3.set\_hobbies("Stunt Kites, Bicycles, Geocaching")

people = [person1, person2, person3]

# Show the common hobbies between person1 and person2

puts "Common hobbies between Alice and Bob: #{person1.common\_hobbies(person2).join(', ')}"

# Find people with similar hobbies to person1

puts "\nPeople with similar hobbies to Alice:"

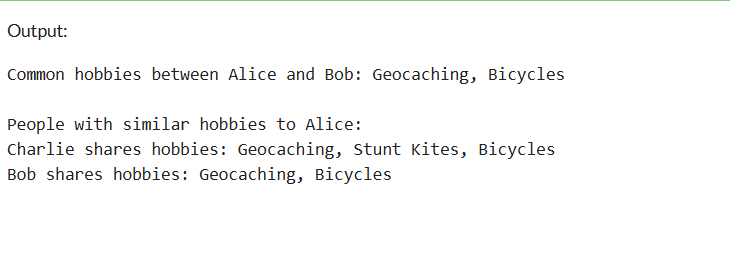
person1.find\_people\_with\_common\_hobbies(people).each do |shared|

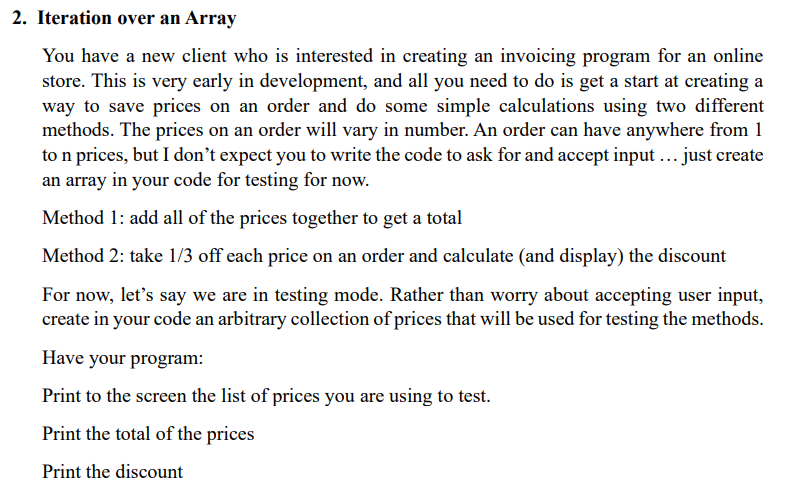
shared.each do |name, hobbies|

puts "#{name} shares hobbies: #{hobbies.join(', ')}"

end

end

****

****

**Implementation :**

**class Order**

**# Method to calculate the total of all prices**

**def initialize(prices)**

**@prices = prices**

**end**

**def total**

**@prices.sum**

**end**

**# Method to calculate the discount (1/3 off each price)**

**def discount**

**@prices.map { |price| price / 3.0 }**

**end**

**# Display the prices, total, and discount**

**def display\_order\_info**

**puts "Prices for the order: #{@prices.join(', ')}"**

**puts "Total: $#{total}"**

**discounted\_prices = discount**

**puts "Discounts (1/3 off each): #{discounted\_prices.map { |price| "$#{'%.2f' % price}" }.join(', ')}"**

**end**

**end**

**# Create an array of prices for testing**

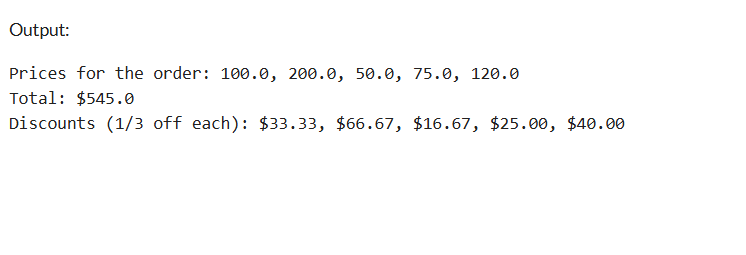
**prices = [100.0, 200.0, 50.0, 75.0, 120.0]**

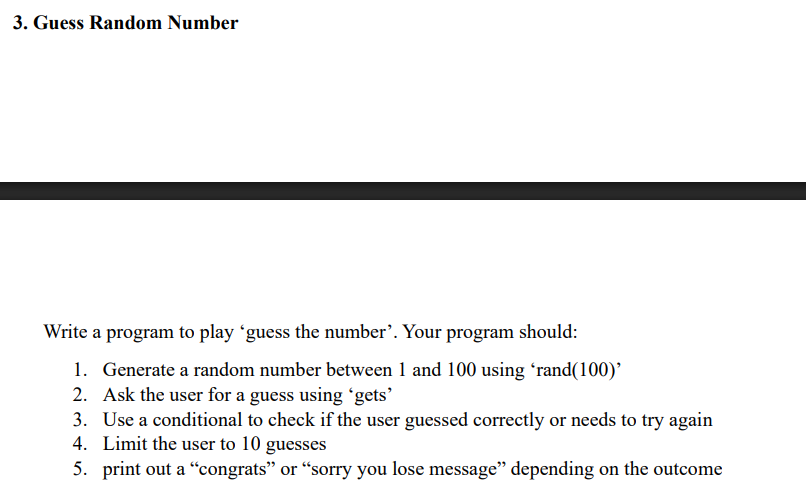
**# Create an Order object with the test prices**

**order = Order.new(prices)**

**# Display the order information**

**order.display\_order\_info**

****

****

**Implementation :**

**class GuessTheNumber**

**def initialize**

**@number\_to\_guess = rand(1..100) # Random number between 1 and 100**

**@max\_attempts = 10**

**@attempts = 0**

**end**

**def play**

**puts "Welcome to Guess the Number!"**

**puts "I'm thinking of a number between 1 and 100."**

**puts "You have #{@max\_attempts} attempts to guess it."**

**while @attempts < @max\_attempts**

**print "Enter your guess (attempt #{@attempts + 1}/#{@max\_attempts}): "**

**guess = gets.to\_i**

**@attempts += 1**

**if guess == @number\_to\_guess**

**puts "Congrats! You guessed the correct number #{@number\_to\_guess}!"**

**break**

**elsif guess < @number\_to\_guess**

**puts "Too low! Try again."**

**else**

**puts "Too high! Try again."**

**end**

**end**

**if @attempts == @max\_attempts && guess != @number\_to\_guess**

**puts "Sorry, you lose! The correct number was #{@number\_to\_guess}."**

**end**

**end**

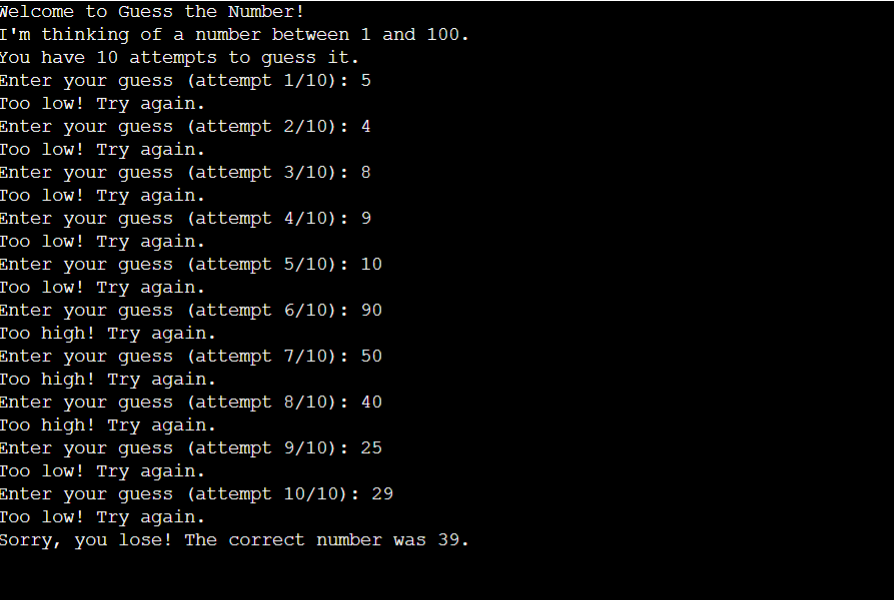
**end**

**# Start the game**

**game = GuessTheNumber.new**

**game.play**

**Output :**





**Implementation :**

# Superclass Media

class Media

attr\_reader :comments

def initialize

@comments = []

end

# Method to add a comment

def add\_comment(comment)

@comments.push(comment)

end

# Method to "play" media

def play

puts "Playing"

end

end

# Subclass Video

class Video < Media

attr\_accessor :resolution

def initialize(resolution)

super() # Initialize the superclass (Media)

@resolution = resolution

end

# Overriding the play method for Video

def play

puts "Playing video at #{@resolution} resolution"

end

end

# Subclass Song

class Song < Media

attr\_accessor :beats\_per\_minute

def initialize(beats\_per\_minute)

super() # Initialize the superclass (Media)

@beats\_per\_minute = beats\_per\_minute

end

# Overriding the play method for Song

def play

puts "Playing song at #{@beats\_per\_minute} beats per minute"

end

end

# Create a video and a song

video = Video.new("1920x1080")

song = Song.new(120)

# Add comments for both video and song

video.add\_comment("Great video! The resolution is stunning!")

video.add\_comment("The music is great in this video too.")

song.add\_comment("Love this song! It's so energetic.")

song.add\_comment("Perfect for workouts.")

# Print out the comments for the video and song

puts "\nVideo Comments:"

video.comments.each { |comment| puts "- #{comment}" }

puts "\nSong Comments:"

song.comments.each { |comment| puts "- #{comment}" }

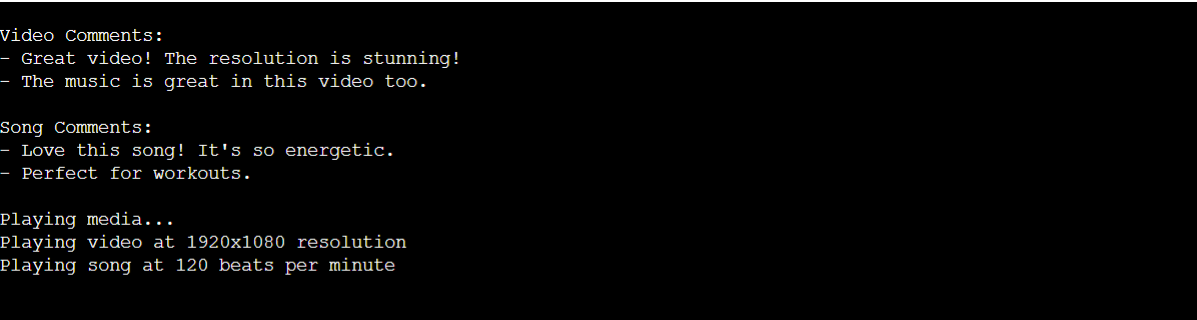
# Play both the video and the song

puts "\nPlaying media..."

video.play

song.play

Output :



**Web Programming - Exercise 4**

****

**Implementation :**

**def calculate\_cart\_total(cart)**

**total\_amount = 0**

**# Iterate over each product in the cart**

**cart.each do |product|**

**product\_name, quantity, price = product**

**# If quantity is greater than 5, apply a 10% discount**

**if quantity > 5**

**price \*= 0.9 # Apply 10% discount**

**end**

**# Add the price of the product (quantity \* price) to the total**

**total\_amount += quantity \* price**

**end**

**return total\_amount**

**end**

**# Example usage:**

**cart = [**

**["Laptop", 2, 1000],**

**["Headphones", 6, 100],**

**["Mouse", 1, 50]**

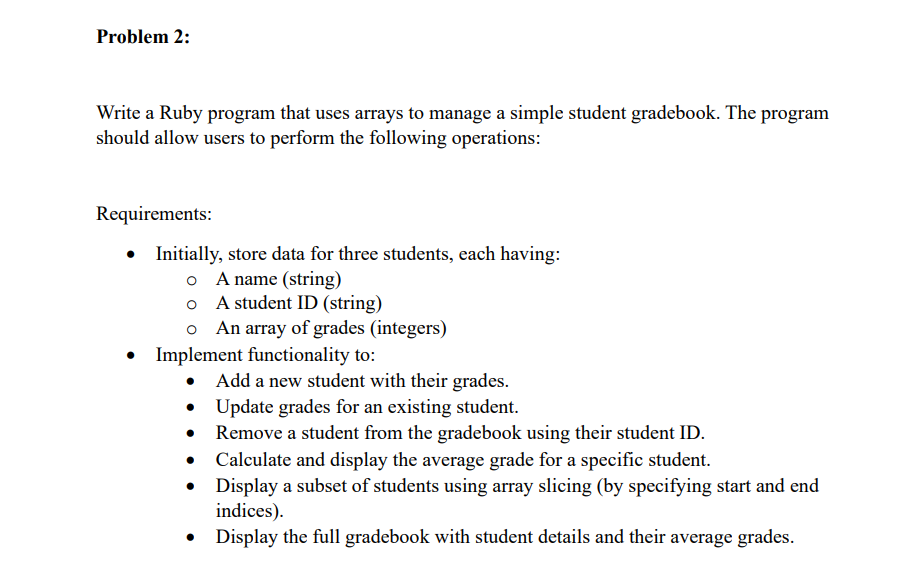
**]**

**final\_amount = calculate\_cart\_total(cart)**

**puts "Final cart total: $#{'%.2f' % final\_amount}"**

**Output :**

****

****

**Implementation :**

**# Initial student data**

**students = [**

**["Alice", "S001", [85, 90, 78]],**

**["Bob", "S002", [75, 80, 88]],**

**["Charlie", "S003", [90, 92, 95]]**

**]**

**# Add a new student**

**def add\_student(students)**

**puts "Enter the new student's name:"**

**name = gets.chomp**

**puts "Enter the student's ID:"**

**student\_id = gets.chomp**

**puts "Enter the student's grades (space-separated):"**

**grades = gets.chomp.split.map(&:to\_i)**

**students << [name, student\_id, grades]**

**puts "New student added!"**

**end**

**# Update grades for an existing student**

**def update\_grades(students)**

**puts "Enter the student ID to update grades:"**

**student\_id = gets.chomp**

**student = students.find { |s| s[1] == student\_id }**

**if student**

**puts "Enter the new grades (space-separated):"**

**new\_grades = gets.chomp.split.map(&:to\_i)**

**student[2] = new\_grades**

**puts "Student grades updated!"**

**else**

**puts "Student not found!"**

**end**

**end**

**# Remove a student by their student ID**

**def remove\_student(students)**

**puts "Enter the student ID to remove:"**

**student\_id = gets.chomp**

**student = students.find { |s| s[1] == student\_id }**

**if student**

**students.delete(student)**

**puts "Removed student with ID: #{student\_id}"**

**puts "Student removed successfully!"**

**else**

**puts "Student not found!"**

**end**

**end**

**# Calculate and display the average grade for a specific student**

**def calculate\_average\_grade(students)**

**puts "Enter the student ID to calculate average grade:"**

**student\_id = gets.chomp**

**student = students.find { |s| s[1] == student\_id }**

**if student**

**average = student[2].sum.to\_f / student[2].size**

**puts "Average grade for #{student[0]}: %.2f" % average**

**else**

**puts "Student not found!"**

**end**

**end**

**# Display a subset of students using array slicing**

**def display\_students\_slice(students)**

**puts "Enter the starting and ending index of students you want to view (space-separated):"**

**start\_index, end\_index = gets.chomp.split.map(&:to\_i)**

**sliced\_students = students[start\_index..end\_index]**

**sliced\_students.each do |student|**

**average = student[2].sum.to\_f / student[2].size**

**puts "#{student[0]} (ID: #{student[1]}) - Average Grade: %.2f" % average**

**end**

**end**

**# Display the full gradebook with student details and average grades**

**def display\_full\_gradebook(students)**

**puts "Gradebook:"**

**puts "Name Student ID Grades Average Grade"**

**students.each do |student|**

**average = student[2].sum.to\_f / student[2].size**

**grades\_str = student[2].join(", ")**

**puts "#{student[0]} #{student[1]} #{grades\_str} %.2f" % average**

**end**

**end**

**# Main menu for interacting with the gradebook**

**loop do**

**puts "\nChoose an option:"**

**puts "1. Add a new student"**

**puts "2. Update grades for an existing student"**

**puts "3. Remove a student by student ID"**

**puts "4. Calculate average grade for a student"**

**puts "5. Display a subset of students"**

**puts "6. Display the full gradebook"**

**puts "7. Exit"**

**choice = gets.chomp.to\_i**

**case choice**

**when 1**

**add\_student(students)**

**when 2**

**update\_grades(students)**

**when 3**

**remove\_student(students)**

**when 4**

**calculate\_average\_grade(students)**

**when 5**

**display\_students\_slice(students)**

**when 6**

**display\_full\_gradebook(students)**

**when 7**

**puts "Exiting program."**

**break**

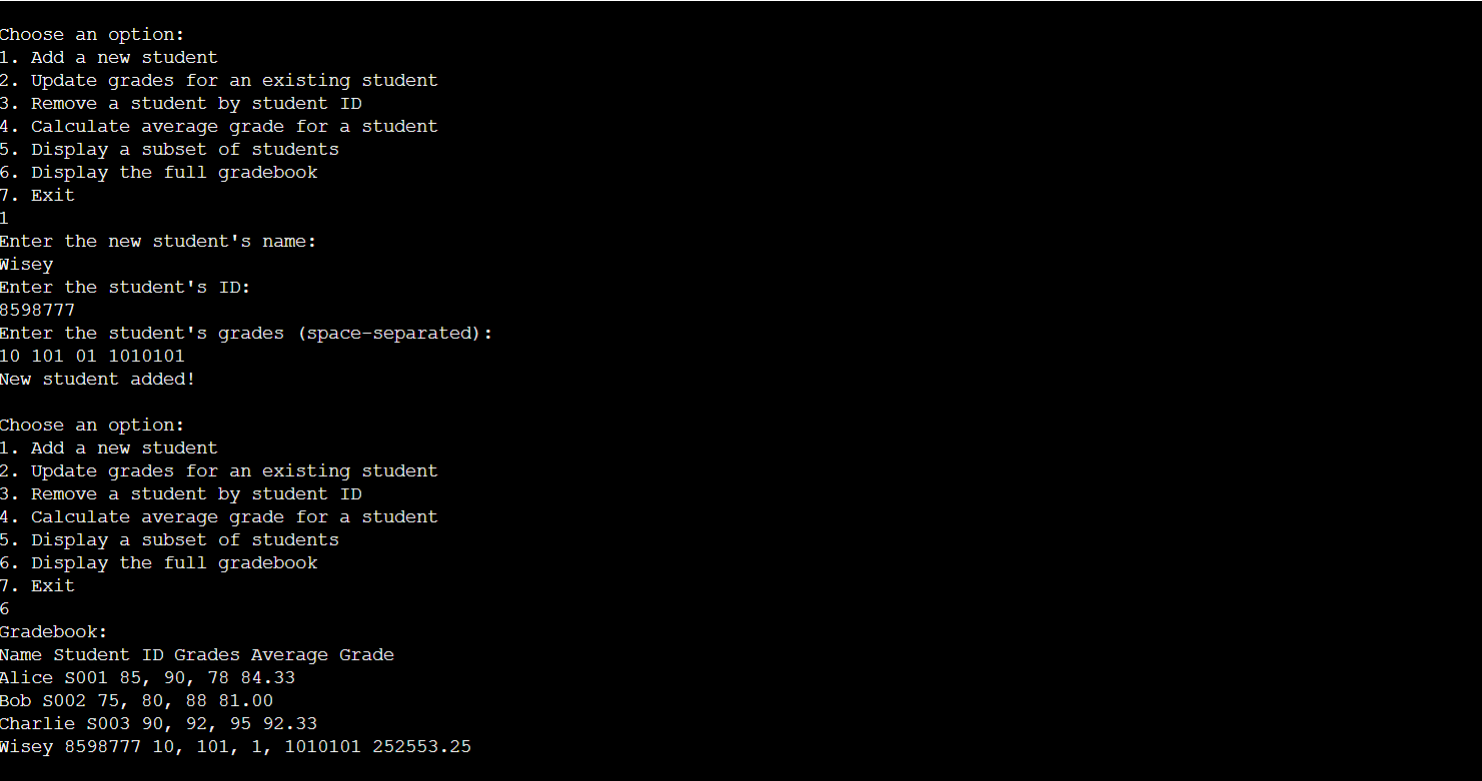
**else**

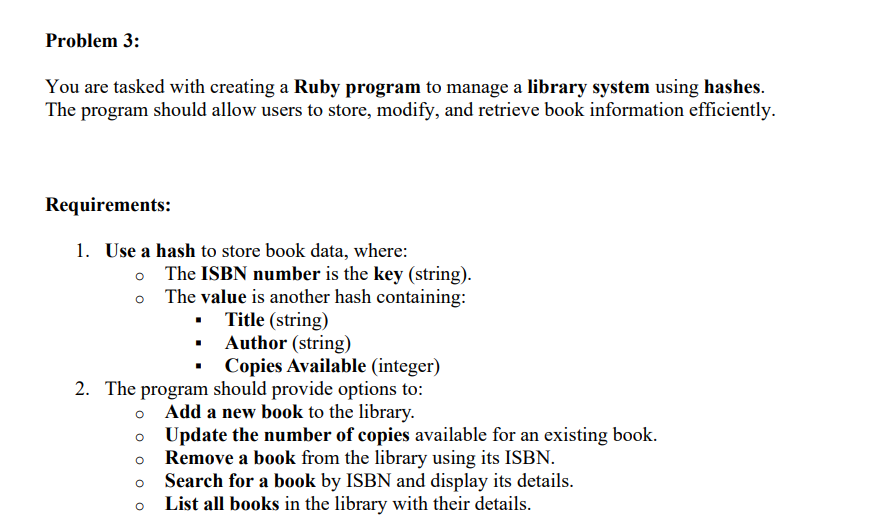
**puts "Invalid choice, please try again."**

**end**

**end**

**Output :**

****

****

**Implementation :**

**# Superclass Media**

**class Media**

**attr\_reader :comments**

**def initialize**

**@comments = []**

**end**

**# Method to add a comment**

**def add\_comment(comment)**

**@comments.push(comment)**

**end**

**# Method to "play" media**

**def play**

**puts "Playing"**

**end**

**end**

**# Subclass Video**

**class Video < Media**

**attr\_accessor :resolution**

**def initialize(resolution)**

**super() # Initialize the superclass (Media)**

**@resolution = resolution**

**end**

**# Overriding the play method for Video**

**def play**

**puts "Playing video at #{@resolution} resolution"**

**end**

**end**

**# Subclass Song**

**class Song < Media**

**attr\_accessor :beats\_per\_minute**

**def initialize(beats\_per\_minute)**

**super() # Initialize the superclass (Media)**

**@beats\_per\_minute = beats\_per\_minute**

**end**

**# Overriding the play method for Song**

**def play**

**puts "Playing song at #{@beats\_per\_minute} beats per minute"**

**end**

**end**

**# Create a video and a song**

**video = Video.new("1920x1080")**

**song = Song.new(120)**

**# Add comments for both video and song**

**video.add\_comment("Great video! The resolution is stunning!")**

**video.add\_comment("The music is great in this video too.")**

**song.add\_comment("Love this song! It's so energetic.")**

**song.add\_comment("Perfect for workouts.")**

**# Print out the comments for the video and song**

**puts "\nVideo Comments:"**

**video.comments.each { |comment| puts "- #{comment}" }**

**puts "\nSong Comments:"**

**song.comments.each { |comment| puts "- #{comment}" }**

**# Play both the video and the song**

**puts "\nPlaying media..."**

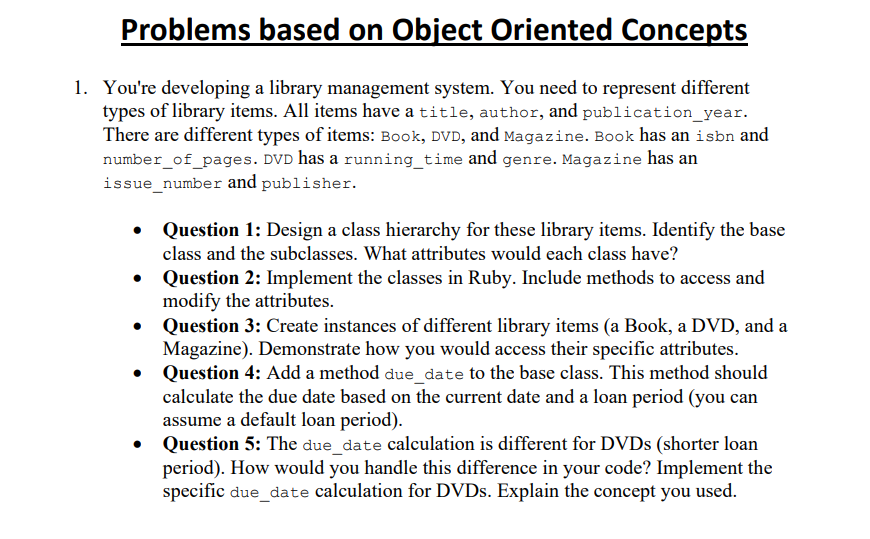
**video.play**

**song.play**

**Output :**

****

**Web Programming - Exercise 5**

****

**Implementations :**

**require 'date'**

**# Base Class: LibraryItem**

**class LibraryItem**

**attr\_accessor :title, :author, :publication\_year**

**def initialize(title, author, publication\_year)**

**@title = title**

**@author = author**

**@publication\_year = publication\_year**

**end**

**# Default loan period (in days)**

**def loan\_period**

**14**

**end**

**# Method to calculate the due date**

**def due\_date**

**today = Date.today**

**due\_date = today + loan\_period**

**due\_date**

**end**

**end**

**# Subclass: Book**

**class Book < LibraryItem**

**attr\_accessor :isbn, :number\_of\_pages**

**def initialize(title, author, publication\_year, isbn, number\_of\_pages)**

**super(title, author, publication\_year)**

**@isbn = isbn**

**@number\_of\_pages = number\_of\_pages**

**end**

**end**

**# Subclass: DVD**

**class DVD < LibraryItem**

**attr\_accessor :running\_time, :genre**

**def initialize(title, author, publication\_year, running\_time, genre)**

**super(title, author, publication\_year)**

**@running\_time = running\_time**

**@genre = genre**

**end**

**# Override the loan\_period method to give DVDs a shorter loan period**

**def loan\_period**

**7**

**end**

**end**

**# Subclass: Magazine**

**class Magazine < LibraryItem**

**attr\_accessor :issue\_number, :publisher**

**def initialize(title, author, publication\_year, issue\_number, publisher)**

**super(title, author, publication\_year)**

**@issue\_number = issue\_number**

**@publisher = publisher**

**end**

**end**

**# Create instances of different library items**

**book = Book.new("The Great Gatsby", "F. Scott Fitzgerald", 1925, "978-0743273565", 180)**

**dvd = DVD.new("Inception", "Christopher Nolan", 2010, 148, "Sci-Fi")**

**magazine = Magazine.new("National Geographic", "Various Authors", 2025, 123, "National Geographic Society")**

**# Access specific attributes**

**puts "Book Title: #{book.title}, ISBN: #{book.isbn}, Pages: #{book.number\_of\_pages}"**

**puts "DVD Title: #{dvd.title}, Genre: #{dvd.genre}, Running Time: #{dvd.running\_time} mins"**

**puts "Magazine Title: #{magazine.title}, Issue Number: #{magazine.issue\_number}, Publisher: #{magazine.publisher}"**

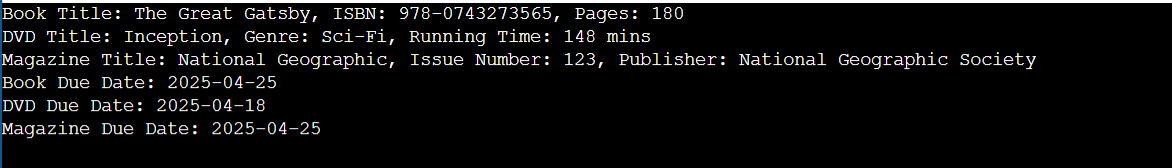
**# Display due dates**

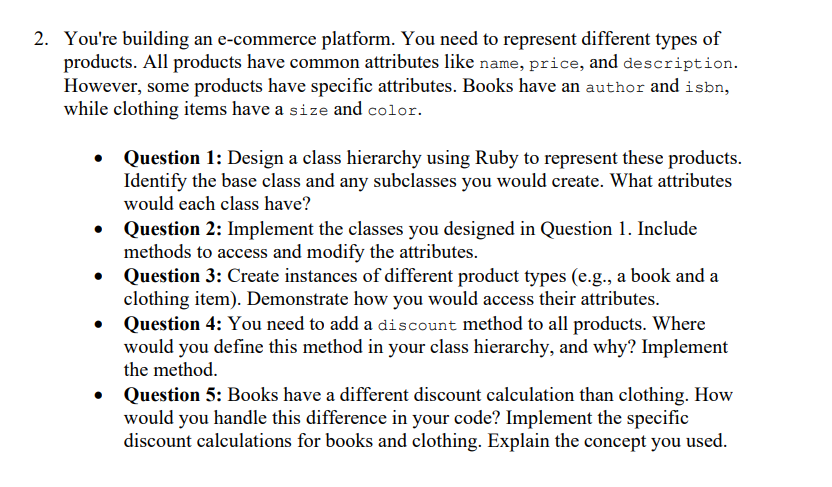
**puts "Book Due Date: #{book.due\_date}"**

**puts "DVD Due Date: #{dvd.due\_date}"**

**puts "Magazine Due Date: #{magazine.due\_date}"**

**Output :**

****

****

**Implementation :**

**# Base Class: Product**

**class Product**

**attr\_accessor :name, :price, :description**

**def initialize(name, price, description)**

**@name = name**

**@price = price**

**@description = description**

**end**

**# Method to get price after applying discount**

**def price\_after\_discount(discount\_percentage)**

**discounted\_price = @price - (@price \* discount\_percentage / 100.0)**

**discounted\_price**

**end**

**end**

**# Subclass: Book**

**class Book < Product**

**attr\_accessor :author, :isbn**

**def initialize(name, price, description, author, isbn)**

**super(name, price, description)**

**@author = author**

**@isbn = isbn**

**end**

**# Override the discount method for books (fixed 10% discount)**

**def price\_after\_discount**

**discount\_percentage = 10**

**discounted\_price = @price - (@price \* discount\_percentage / 100.0)**

**discounted\_price**

**end**

**end**

**# Subclass: Clothing**

**class Clothing < Product**

**attr\_accessor :size, :color**

**def initialize(name, price, description, size, color)**

**super(name, price, description)**

**@size = size**

**@color = color**

**end**

**# Override the discount method for clothing (10% discount if price > $30, 5% otherwise)**

**def price\_after\_discount**

**if @price > 30**

**discount\_percentage = 10**

**else**

**discount\_percentage = 5**

**end**

**discounted\_price = @price - (@price \* discount\_percentage / 100.0)**

**discounted\_price**

**end**

**end**

**# Create instances of Book and Clothing**

**book = Book.new("The Catcher in the Rye", 20.00, "A classic novel by J.D. Salinger", "J.D. Salinger", "978-0-316-76948-0")**

**clothing = Clothing.new("T-shirt", 25.00, "Cotton T-shirt in various sizes", "M", "Blue")**

**# Accessing and displaying attributes**

**puts "Book Name: #{book.name}, Author: #{book.author}, ISBN: #{book.isbn}, Price: $#{book.price}, Description: #{book.description}"**

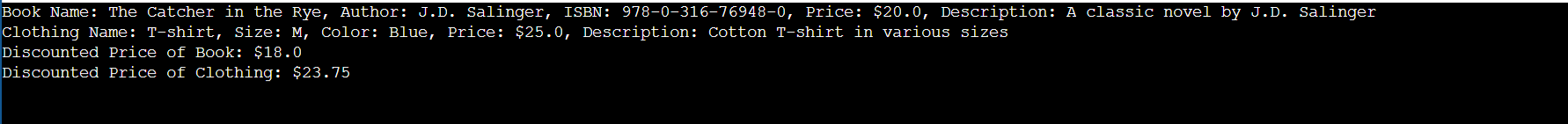
**puts "Clothing Name: #{clothing.name}, Size: #{clothing.size}, Color: #{clothing.color}, Price: $#{clothing.price}, Description: #{clothing.description}"**

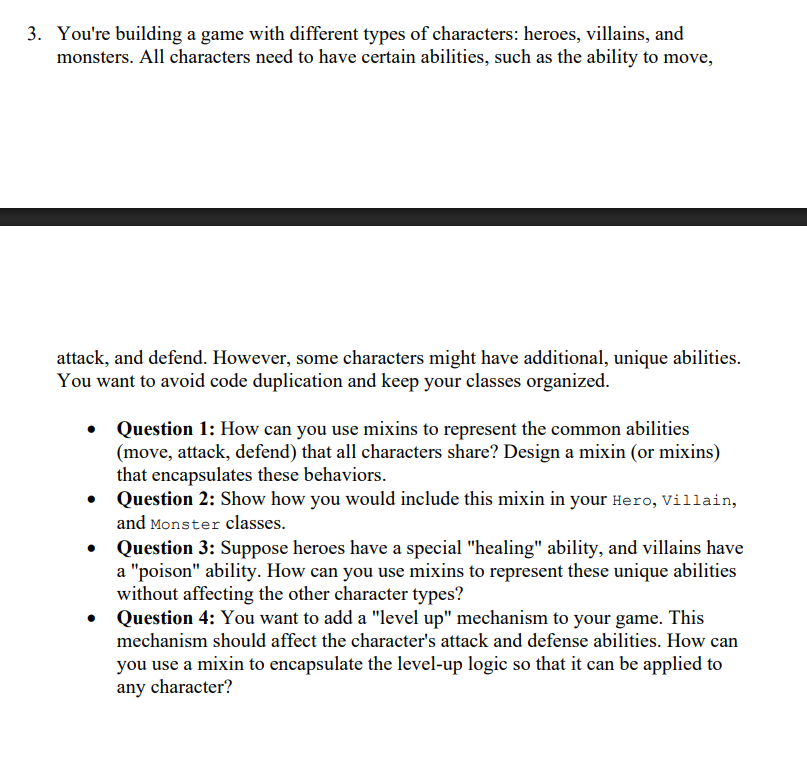
**# Accessing prices after discount**

**puts "Discounted Price of Book: $#{book.price\_after\_discount}" # 10% discount**

**puts "Discounted Price of Clothing: $#{clothing.price\_after\_discount}" # 5% discount (because price is less than $30)**

**Output :**



****

**Implementation :**

**# Define the modules for Abilities, Healing, Poison, and LevelUp**

**module Abilities**

**def move**

**puts "#{self.class} moves to a new position."**

**end**

**def attack**

**puts "#{self.class} attacks the enemy!"**

**end**

**def defend**

**puts "#{self.class} defends against the attack!"**

**end**

**end**

**module Healing**

**def heal**

**puts "#{self.class} uses healing powers to restore health!"**

**end**

**end**

**module Poison**

**def poison**

**puts "#{self.class} poisons the enemy, causing damage over time!"**

**end**

**end**

**module LevelUp**

**def level\_up**

**@attack += 10**

**@defense += 5**

**puts "#{self.class} leveled up! Attack: #{@attack}, Defense: #{@defense}"**

**end**

**end**

**# Now define the classes**

**class Hero**

**include Abilities**

**include Healing**

**include LevelUp**

**def initialize(name, attack = 20, defense = 15)**

**@name = name**

**@attack = attack**

**@defense = defense**

**end**

**def display**

**puts "Hero #{@name} is ready for action!"**

**end**

**end**

**class Villain**

**include Abilities**

**include Poison**

**include LevelUp**

**def initialize(name, attack = 25, defense = 10)**

**@name = name**

**@attack = attack**

**@defense = defense**

**end**

**def display**

**puts "Villain #{@name} is ready to cause trouble!"**

**end**

**end**

**class Monster**

**include Abilities**

**include LevelUp**

**def initialize(name, attack = 30, defense = 20)**

**@name = name**

**@attack = attack**

**@defense = defense**

**end**

**def display**

**puts "Monster #{@name} is ready to fight!"**

**end**

**end**

**# Create instances and test the functionality**

**hero = Hero.new("Knight")**

**villain = Villain.new("Sorcerer")**

**monster = Monster.new("Dragon")**

**hero.display**

**hero.move**

**hero.attack**

**hero.defend**

**hero.heal**

**hero.level\_up**

**villain.display**

**villain.move**

**villain.attack**

**villain.defend**

**villain.poison**

**villain.level\_up**

**monster.display**

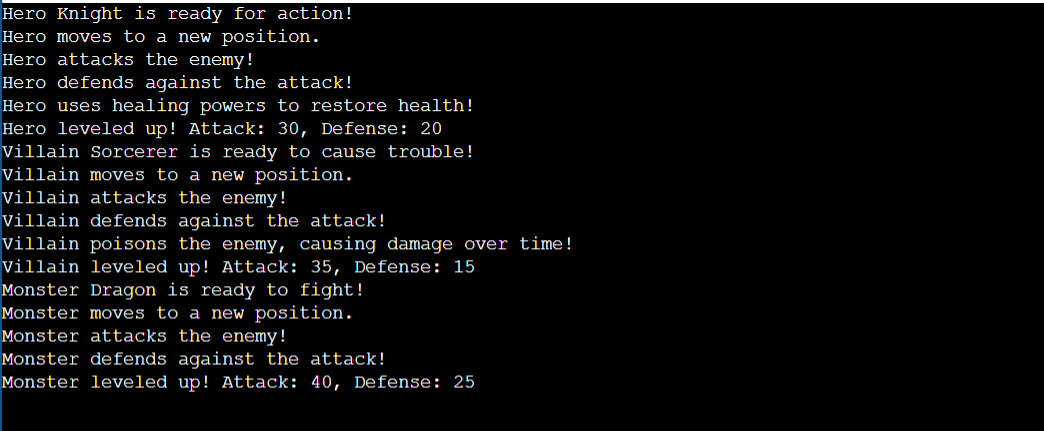
**monster.move**

**monster.attack**

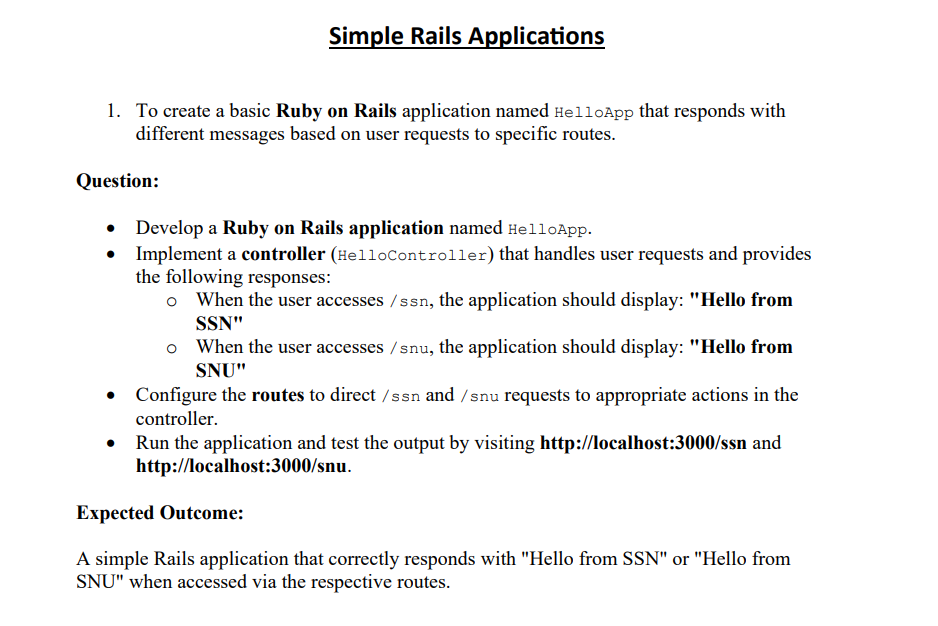
**monster.defend**

**monster.level\_up**

**Output :**

****

**Web Programming - Exercise 6**

****

1.rails new HelloApp

2.cd HelloApp

3.rails generate controller Hello

4. app/controllers/hello\_controller.rb

**Implementation** :

class HelloController < ApplicationController

def ssn

render plain: "Hello from SSN"

end

def snu

render plain: "Hello from SNU"

end

end

5.config/routes.rb

Rails.application.routes.draw do

get '/ssn', to: 'hello#ssn'

get '/snu', to: 'hello#snu'

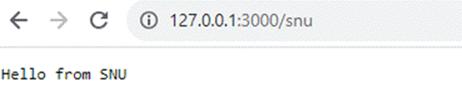
# You can leave the root route as is or set it to a default page

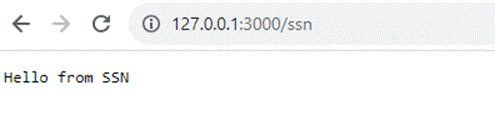
# root 'hello#index'

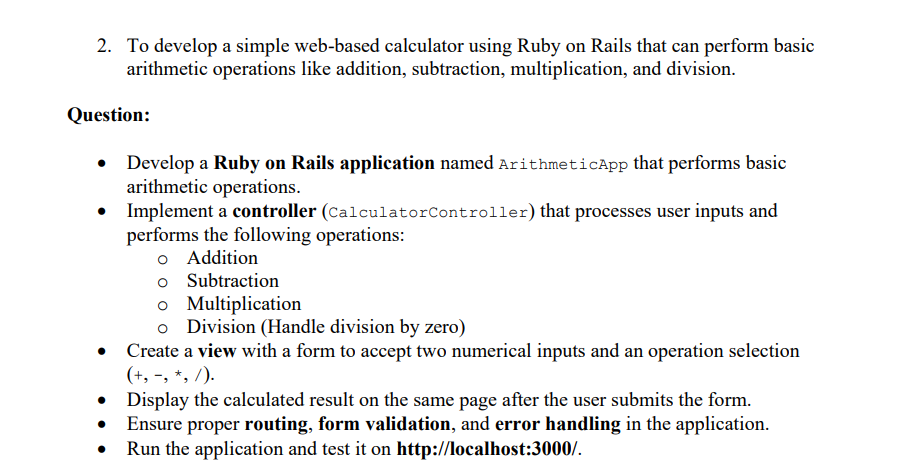
End

6.rails server

**OUTPUT:**

****

****

****

1.rails new ArithmeticApp

2.cd ArithmeticApp

3.rails generate controller Calculator

**Implementation :**

config/routes.rb

Rails.application.routes.draw do

root 'calculator#index' # Set the root route to show the form

get 'calculator/index' # Route to render the form

post 'calculator/calculate' # Route to handle form submission

end

app/controllers/calculator\_controller.rb

class CalculatorController < ApplicationController

def index

# This action renders the form on the first page load.

end

def calculate

# Get the input values from the form

num1 = params[:num1].to\_f

num2 = params[:num2].to\_f

operation = params[:operation]

# Perform the operation based on the user's input

case operation

when "+"

@result = num1 + num2

when "-"

@result = num1 - num2

when "\*"

@result = num1 \* num2

when "/"

if num2 == 0

@result = "Error: Division by zero"

else

@result = num1 / num2

end

else

@result = "Invalid operation"

end

# Render the form again with the result (no page redirect, just render same page)

render :index

end

end

app/views/calculator/index.html.erb

<h1>Simple Calculator</h1>

<%= form\_with url: '/calculator/calculate', method: :post, data: { turbo: false } do %>

<div>

<label for="num1">Number 1:</label>

<input type="number" name="num1" required>

</div>

<div>

<label for="num2">Number 2:</label>

<input type="number" name="num2" required>

</div>

<div>

<label for="operation">Operation:</label>

<select name="operation" required>

<option value="+">Addition (+)</option>

<option value="-">Subtraction (-)</option>

<option value="\*">Multiplication (\*)</option>

<option value="/">Division (/)</option>

</select>

</div>

<div>

<button type="submit">Calculate</button>

</div>

<% end %>

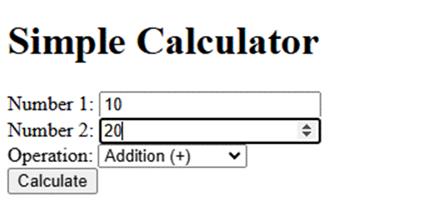
<% if defined?(@result) %>

<h2>Result: <%= @result %></h2>

<% end %>

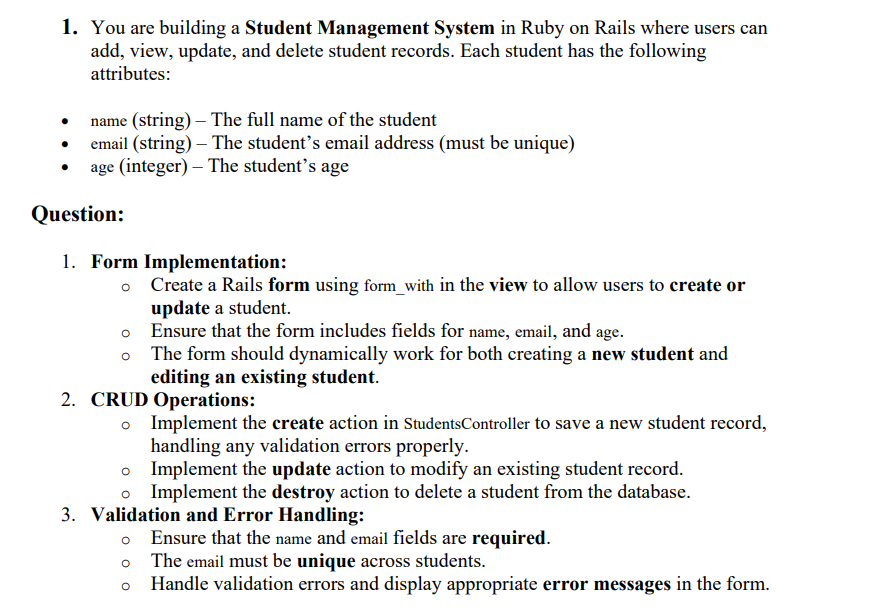
4.rails server

Output :





**Web Programming - Exercise 7**

****

**Implementation :**

**Code:**

**app/views/books/\_form.html.erb**

**<%= form\_with(model: book, local: true) do |form| %>**

**<% if book.errors.any? %>**

**<div id="error\_explanation">**

**<h2><%= pluralize(book.errors.count, "error") %> prohibited this book from being saved:</h2>**

**<ul>**

**<% book.errors.full\_messages.each do |message| %>**

**<li><%= message %></li>**

**<% end %>**

**</ul>**

**</div>**

**<% end %>**

**<div class="field">**

**<%= form.label :title %><br>**

**<%= form.text\_field :title %>**

**</div>**

**<div class="field">**

**<%= form.label :author %><br>**

**<%= form.text\_field :author %>**

**</div>**

**<div class="field">**

**<%= form.label :published\_year %><br>**

**<%= form.number\_field :published\_year %>**

**</div>**

**<div class="actions">**

**<%= form.submit %>**

**</div>**

**<% end %>**

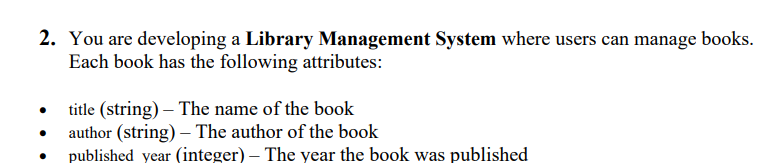
**<%= render 'form', book: @book %>**

**Output :**

****

****

****

****

**Implementation :**

**class BooksController < ApplicationController**

**before\_action :set\_book, only: %i[show edit update destroy]**

**def index**

**@books = Book.all**

**end**

**def show**

**end**

**def new**

**@book = Book.new**

**end**

**def create**

**@book = Book.new(book\_params)**

**if @book.save**

**redirect\_to @book, notice: 'Book was successfully created.'**

**else**

**render :new, status: :unprocessable\_entity**

**end**

**end**

**def edit**

**end**

**def update**

**if @book.update(book\_params)**

**redirect\_to @book, notice: 'Book was successfully updated.'**

**else**

**render :edit, status: :unprocessable\_entity**

**end**

**end**

**def destroy**

**@book.destroy**

**redirect\_to books\_path, notice: 'Book was successfully deleted.'**

**end**

**private**

**def set\_book**

**@book = Book.find(params[:id])**

**end**

**def book\_params**

**params.require(:book).permit(:title, :author, :published\_year)**

**end**

**end**

****

**Implementation :**

**class Book < ApplicationRecord**

**validates :title, presence: true**

**validates :author, presence: true**

**validates :published\_year, numericality: { only\_integer: true, less\_than\_or\_equal\_to: Date.today.year }**

**end**

**4. Index and Show Pages:**

**o Implement an index page that lists all books with options to edit or delete**

**them.**

**o Implement a show page to display details of a single book.**

**Expected Output:**

**• A form that can be used to add or edit a book.**

**• Complete CRUD functionality implemented in the controller (create, update, destroy).**

**• Proper error messages displayed in case of validation failures.**

**• An index page that lists all books and a show page to view book details.**

**Code:**

**app/views/books/index.html.erb**

**<h1>All Books</h1>**

**<%= link\_to 'Add New Book', new\_book\_path %>**

**<table>**

**<thead>**

**<tr>**

**<th>Title</th>**

**<th>Author</th>**

**<th>Published Year</th>**

**<th>Actions</th>**

**</tr>**

**</thead>**

**<tbody>**

**<% @books.each do |book| %>**

**<tr>**

**<td><%= link\_to book.title, book %></td>**

**<td><%= book.author %></td>**

**<td><%= book.published\_year %></td>**

**<td>**

**<%= link\_to 'Edit', edit\_book\_path(book) %> |**

**<%= link\_to 'Delete', book, method: :delete, data: { confirm: 'Are you sure?' } %>**

**</td>**

**</tr>**

**<% end %>**

**</tbody>**

**</table>**

**app/views/books/show.html.erb**

**<h1><%= @book.title %></h1>**

**<p><strong>Author:</strong> <%= @book.author %></p>**

**<p><strong>Published Year:</strong> <%= @book.published\_year %></p>**

**<%= link\_to 'Edit', edit\_book\_path(@book) %> |**

**<%= link\_to 'Back to List', books\_path %>**

**rails generate model Book title:string author:string published\_year:integer**

**rails db:migrate**

**Output :**

****

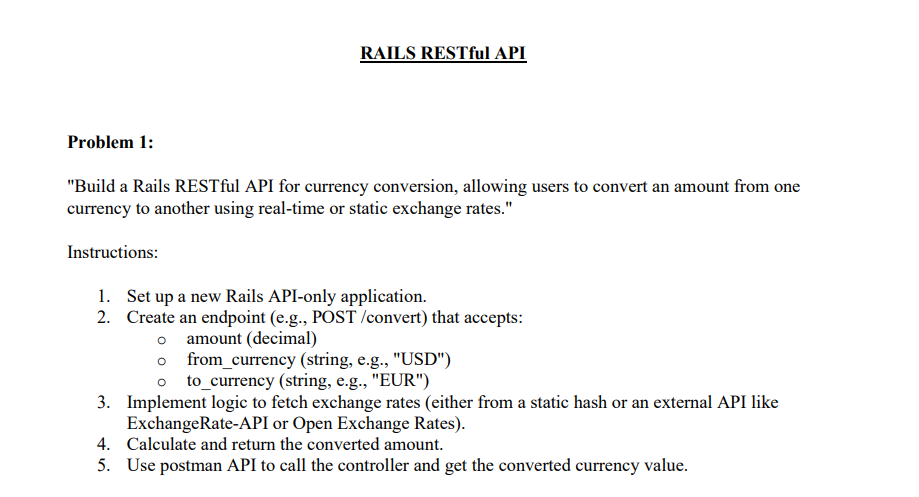
****

****

****

****

**Web Programming - Exercise 8**

****

**Implementation :**

**Step 1: Create API-only Rails App**

**rails new currency\_converter\_api --api**

**cd currency\_converter\_api**

**Step 2: Generate Controller**

**rails generate controller Currency**

**Step 3: Define Your Route**

**Edit config/routes.rb:**

**Rails.application.routes.draw do**

**post 'convert', to: 'currency#convert'**

**End**

**app/controllers/currency\_controller.rb**

**class CurrencyController < ApplicationController**

**STATIC\_RATES = {**

**"USD" => {**

**"EUR" => 0.93,**

**"INR" => 83.12,**

**"GBP" => 0.80**

**},**

**"EUR" => {**

**"USD" => 1.08,**

**"INR" => 89.25,**

**"GBP" => 0.86**

**},**

**"INR" => {**

**"USD" => 0.012,**

**"EUR" => 0.011,**

**"GBP" => 0.0096**

**}**

**# Add more rates as needed**

**}**

**def convert**

**amount = params[:amount].to\_f**

**from = params[:from\_currency]**

**to = params[:to\_currency]**

**rate = STATIC\_RATES.dig(from, to)**

**if rate**

**converted = amount \* rate**

**render json: {**

**from: from,**

**to: to,**

**amount: amount,**

**converted\_amount: converted.round(2),**

**rate: rate**

**}, status: :ok**

**else**

**render json: { error: "Exchange rate not found for #{from} to #{to}" }, status: :unprocessable\_entity**

**end**

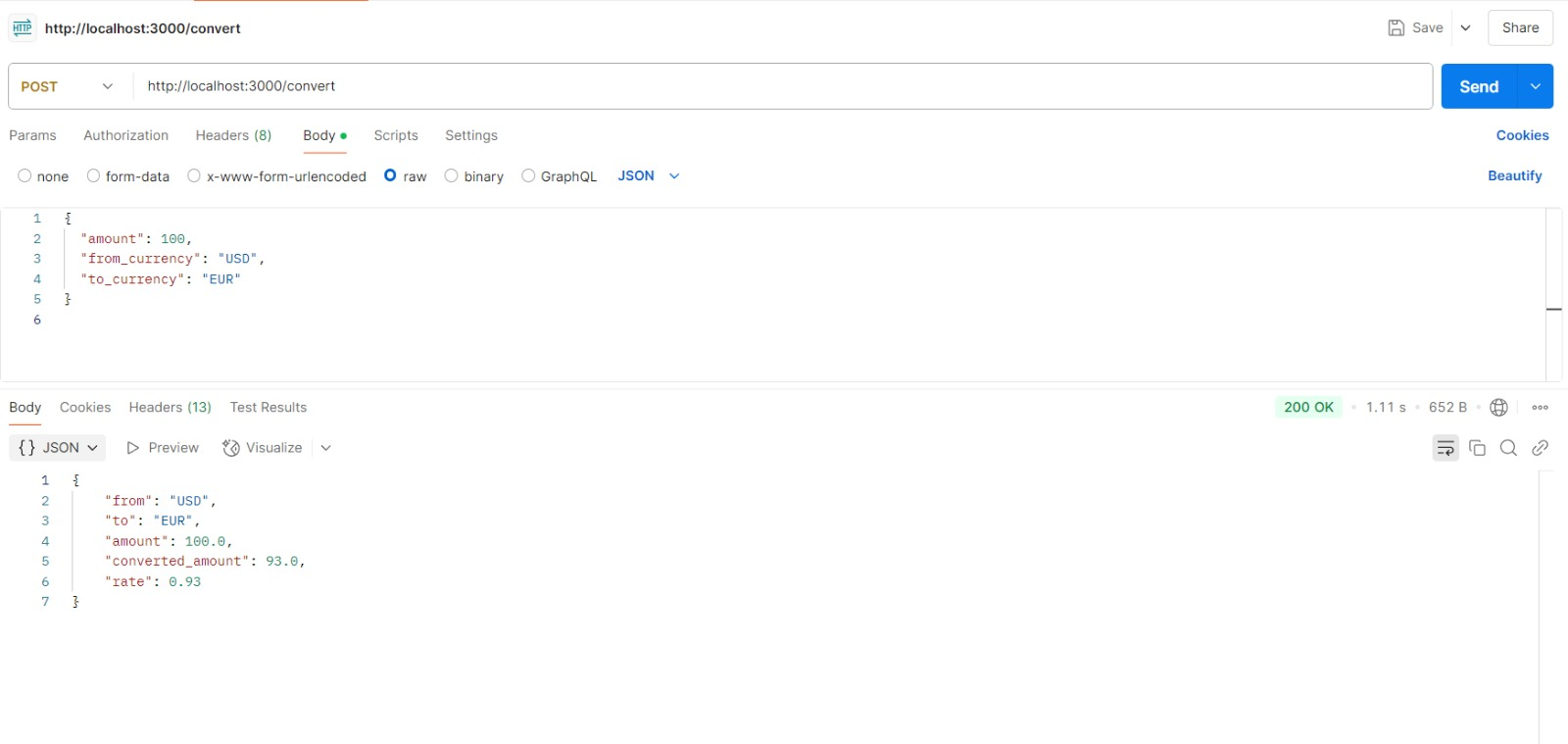
**end**

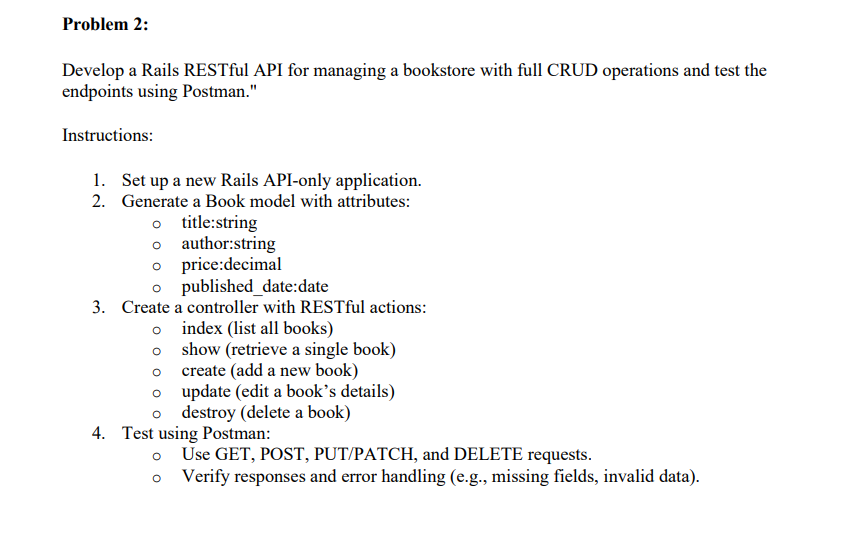
**end**

**Step 4: Test with Postman**

**Use Postman to send a POST request to:**

**http://localhost:3000/convert**

****

****

**Implementation :**

**Develop a Rails RESTful API for managing a bookstore with full CRUD operations and test the**

**endpoints using Postman."**

**Instructions:**

**1. Set up a new Rails API-only application.**

**rails new bookstore\_api --api**

**cd bookstore\_api**

**2. Generate a Book model with attributes:**

**o title:string**

**o author:string**

**o price:decimal**

**o published\_date:date**

**rails generate model Book title:string author:string price:decimal published\_date:date**

**rails db:migrate**

**3. Create a controller with RESTful actions:**

**o index (list all books)**

**o show (retrieve a single book)**

**o create (add a new book)**

**o update (edit a book’s details)**

**o destroy (delete a book)**

**rails generate controller Books**

**config/routes.rb**

**Rails.application.routes.draw do**

**resources :books**

**end**

**4. Test using Postman:**

**o Use GET, POST, PUT/PATCH, and DELETE requests.**

**o Verify responses and error handling (e.g., missing fields, invalid data)**

**app/controllers/books\_controller.rb**

**class BooksController < ApplicationController**

**before\_action :set\_book, only: [:show, :update, :destroy]**

**# GET /books**

**def index**

**books = Book.all**

**render json: books**

**end**

**# GET /books/:id**

**def show**

**render json: @book**

**end**

**# POST /books**

**def create**

**book = Book.new(book\_params)**

**if book.save**

**render json: book, status: :created**

**else**

**render json: { errors: book.errors.full\_messages }, status: :unprocessable\_entity**

**end**

**end**

**# PUT/PATCH /books/:id**

**def update**

**if @book.update(book\_params)**

**render json: @book**

**else**

**render json: { errors: @book.errors.full\_messages }, status: :unprocessable\_entity**

**end**

**end**

**# DELETE /books/:id**

**def destroy**

**@book.destroy**

**render json: { message: "Book deleted successfully" }**

**end**

**private**

**def set\_book**

**@book = Book.find\_by(id: params[:id])**

**render json: { error: "Book not found" }, status: :not\_found unless @book**

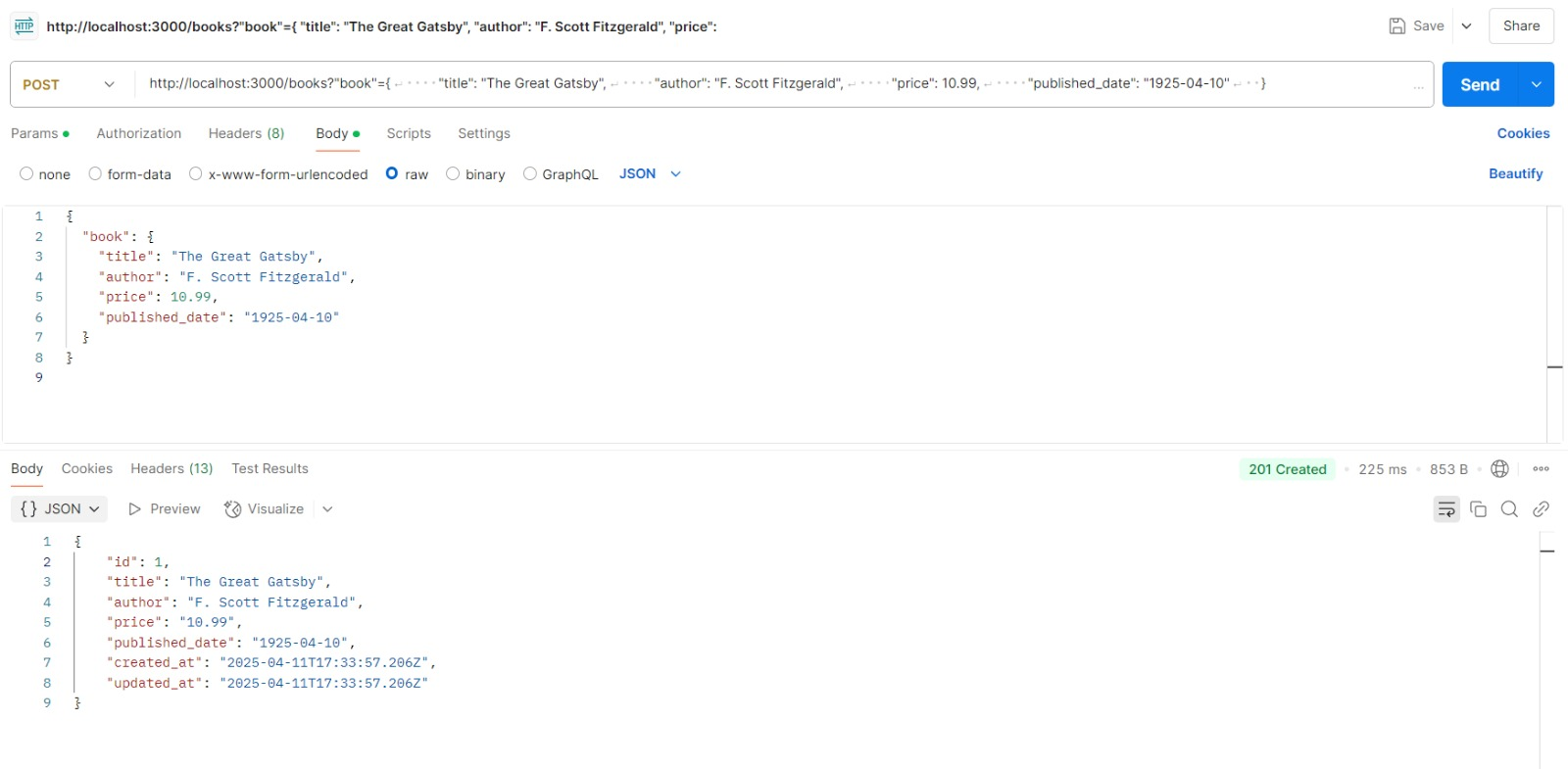
**end**

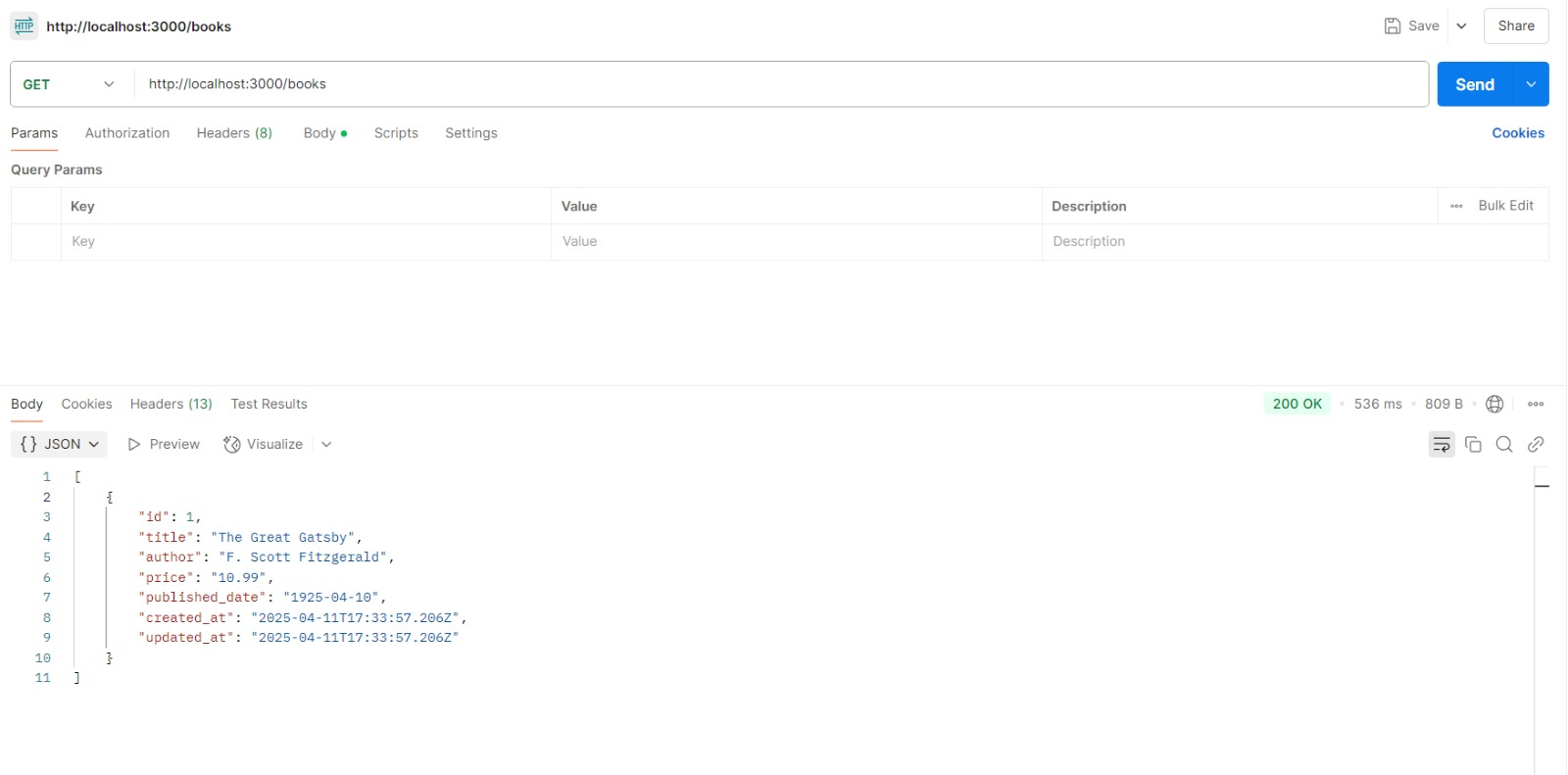
**def book\_params**

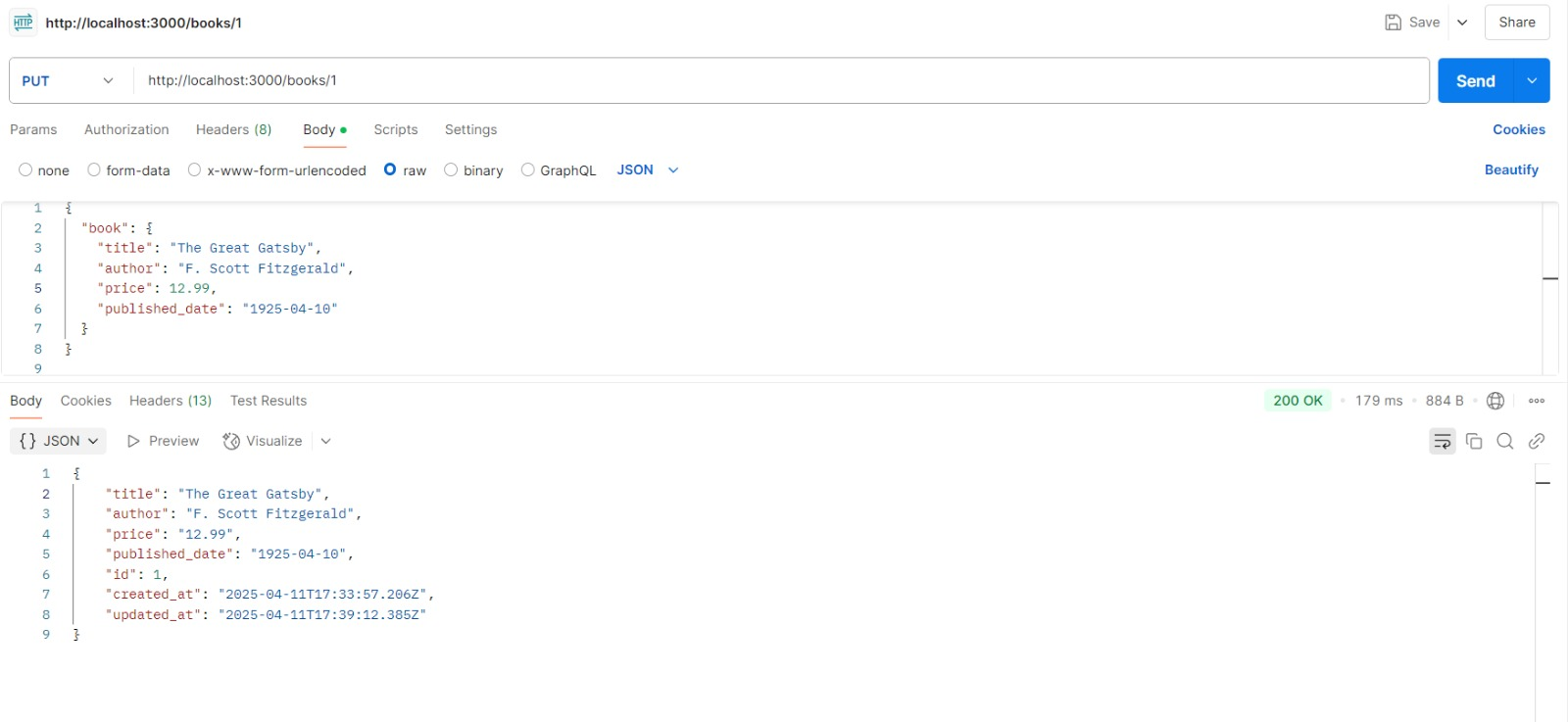
**params.require(:book).permit(:title, :author, :price, :published\_date)**

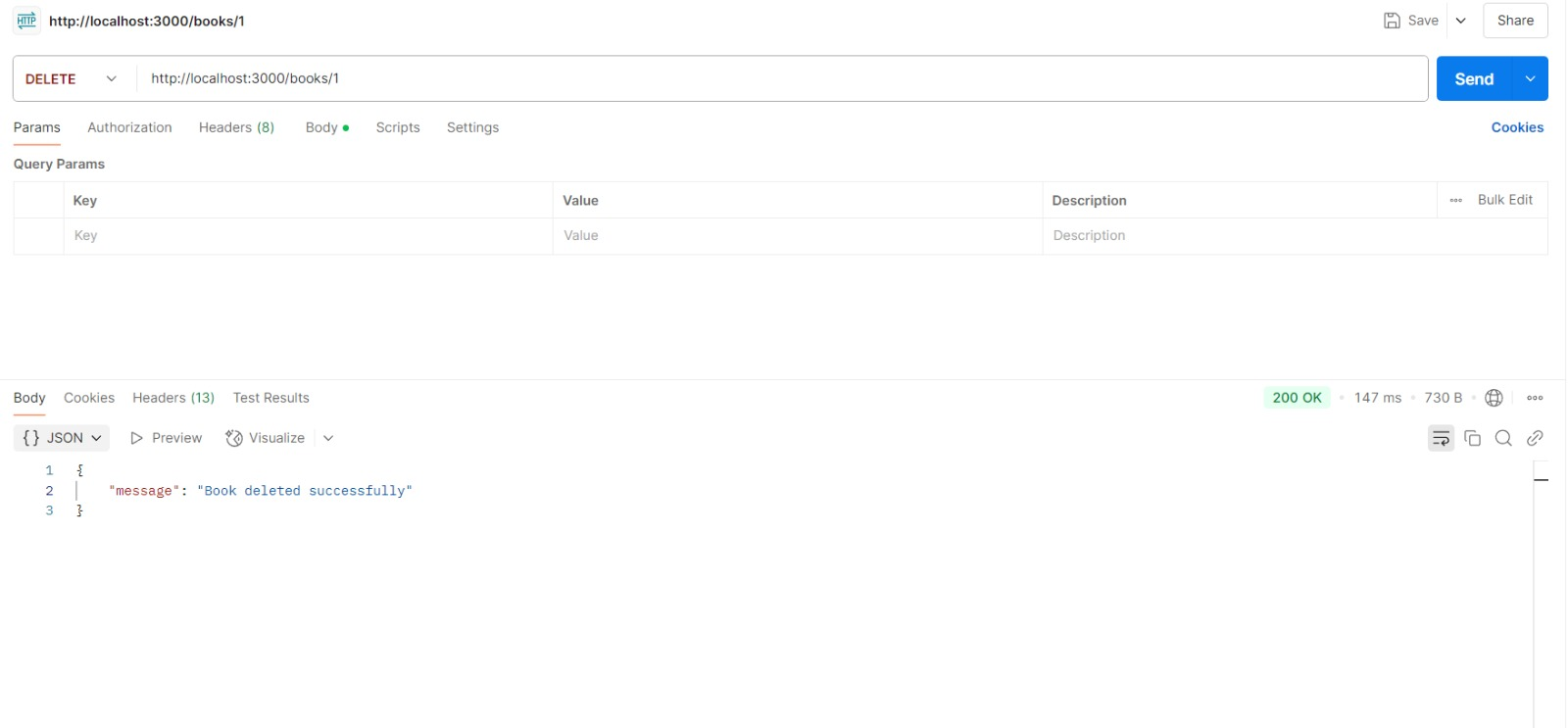
**end**

**End**

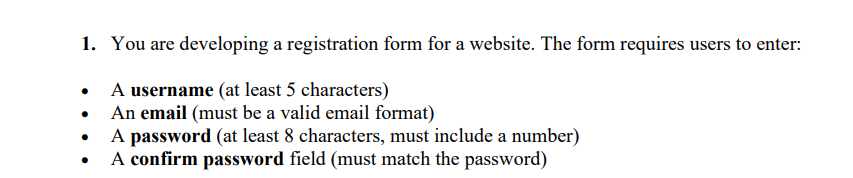
****

****

****

****

**Web Programming - Exercise 9**

****

**Implementation :**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>Registration Form</title>**

**<style>**

**body {**

**font-family: Arial, sans-serif;**

**margin: 40px;**

**background-color: #f4f4f4;**

**}**

**form {**

**background: #fff;**

**padding: 20px;**

**max-width: 400px;**

**border-radius: 8px;**

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

**}**

**label {**

**display: block;**

**margin-top: 10px;**

**}**

**input[type="text"],**

**input[type="password"] {**

**width: 100%;**

**padding: 8px;**

**margin-top: 4px;**

**border: 1px solid #ccc;**

**border-radius: 4px;**

**}**

**button {**

**margin-top: 15px;**

**padding: 10px 20px;**

**background-color: #0E4AA0;**

**color: white;**

**border: none;**

**border-radius: 4px;**

**cursor: pointer;**

**}**

**#errorMessages {**

**color: red;**

**margin-top: 15px;**

**}**

**</style>**

**</head>**

**<body>**

**<h2>Registration Form</h2>**

**<form id="registrationForm" onsubmit="return validateForm()">**

**<label for="username">Username:</label>**

**<input type="text" id="username" />**

**<label for="email">Email:</label>**

**<input type="text" id="email" />**

**<label for="password">Password:</label>**

**<input type="password" id="password" />**

**<label for="confirmPassword">Confirm Password:</label>**

**<input type="password" id="confirmPassword" />**

**<button type="submit">Register</button>**

**<div id="errorMessages"></div>**

**</form>**

**<script>**

**function validateForm() {**

**const username = document.getElementById("username").value.trim();**

**const email = document.getElementById("email").value.trim();**

**const password = document.getElementById("password").value;**

**const confirmPassword = document.getElementById("confirmPassword").value;**

**const errorDiv = document.getElementById("errorMessages");**

**errorDiv.innerHTML = ""; // Clear previous errors**

**let errors = [];**

**// Username validation**

**if (username.length < 5) {**

**errors.push("Username must be at least 5 characters long.");**

**}**

**// Email validation using regex**

**const emailPattern = /^[^\\s@]+@[^\\s@]+\\.[^\\s@]+$/;**

**if (!emailPattern.test(email)) {**

**errors.push("Please enter a valid email address.");**

**}**

**// Password validation**

**if (password.length < 8) {**

**errors.push("Password must be at least 8 characters long.");**

**} else if (!/\d/.test(password)) {**

**errors.push("Password must include at least one number.");**

**}**

**// Confirm password**

**if (password !== confirmPassword) {**

**errors.push("Passwords do not match.");**

**}**

**// Show errors or allow form submission**

**if (errors.length > 0) {**

**errorDiv.innerHTML = errors.join("<br>");**

**return false; // Prevent form submission**

**}**

**alert("Form submitted successfully!");**

**return true; // Allow form submission**

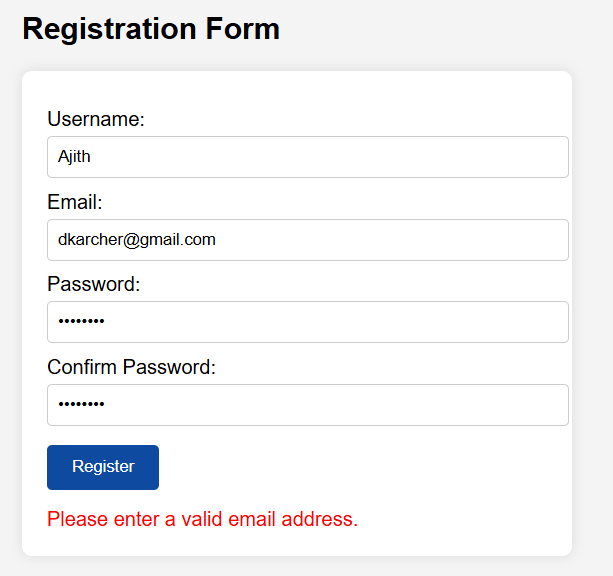
**}**

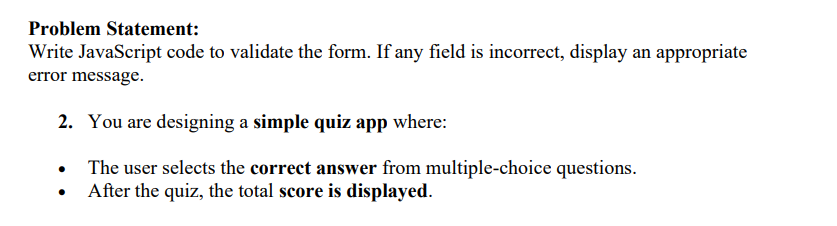
**</script>**

**</body>**

**</html>**

**Output :**

****

****

**Implementation :**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>Simple Quiz App</title>**

**<style>**

**body {**

**font-family: Arial, sans-serif;**

**margin: 40px;**

**background-color: #f0f0f0;**

**}**

**#quizContainer {**

**background: #fff;**

**padding: 20px;**

**max-width: 500px;**

**border-radius: 8px;**

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

**}**

**h2 {**

**margin-bottom: 10px;**

**}**

**.option {**

**margin: 8px 0;**

**}**

**button {**

**margin-top: 15px;**

**padding: 10px 20px;**

**background-color: #0E4AA0;**

**color: white;**

**border: none;**

**border-radius: 5px;**

**cursor: pointer;**

**}**

**#score {**

**margin-top: 20px;**

**font-size: 18px;**

**font-weight: bold;**

**color: green;**

**}**

**</style>**

**</head>**

**<body>**

**<div id="quizContainer">**

**<h2 id="question">Question text</h2>**

**<div id="options"></div>**

**<button onclick="submitAnswer()">Submit</button>**

**<div id="score"></div>**

**</div>**

**<script>**

**const quiz = [**

**{**

**question: "What is the capital of France?",**

**options: ["Berlin", "Madrid", "Paris", "Rome"],**

**answer: "Paris"**

**},**

**{**

**question: "What is 2 + 2?",**

**options: ["3", "4", "5", "6"],**

**answer: "4"**

**},**

**{**

**question: "Which language runs in a web browser?",**

**options: ["Java", "C", "Python", "JavaScript"],**

**answer: "JavaScript"**

**}**

**];**

**let currentQuestion = 0;**

**let score = 0;**

**function loadQuestion() {**

**const q = quiz[currentQuestion];**

**document.getElementById("question").textContent = q.question;**

**const optionsDiv = document.getElementById("options");**

**optionsDiv.innerHTML = "";**

**q.options.forEach(option => {**

**const optionHTML = `**

**<div class="option">**

**<label>**

**<input type="radio" name="answer" value="${option}" /> ${option}**

**</label>**

**</div>**

**`;**

**optionsDiv.innerHTML += optionHTML;**

**});**

**}**

**function submitAnswer() {**

**const selected = document.querySelector('input[name="answer"]:checked');**

**if (!selected) {**

**alert("Please select an answer.");**

**return;**

**}**

**const answer = selected.value;**

**if (answer === quiz[currentQuestion].answer) {**

**score++;**

**}**

**currentQuestion++;**

**if (currentQuestion < quiz.length) {**

**loadQuestion();**

**} else {**

**document.getElementById("quizContainer").innerHTML = `**

**<h2>Quiz Completed!</h2>**

**<p id="score">Your score: ${score} out of ${quiz.length}</p>**

**`;**

**}**

**}**

**// Initial load**

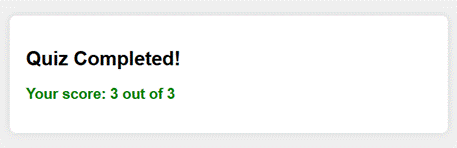
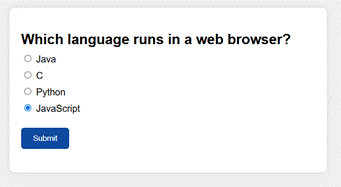
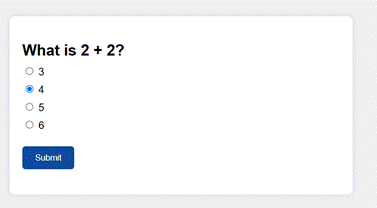
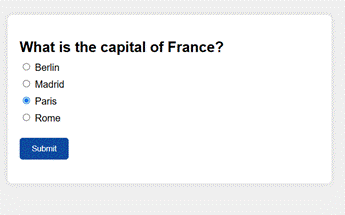
**loadQuestion();**

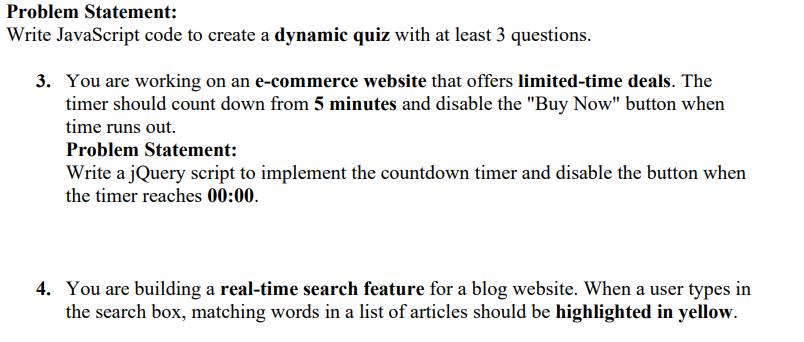
**</script>**

**</body>**

**</html>**

**Output:**

****

****

**Implementation :**

**<title>Limited Time Deal</title>**

**<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>**

**<style>**

**body {**

**font-family: Arial, sans-serif;**

**padding: 40px;**

**background-color: #f9f9f9;**

**}**

**#timer {**

**font-size: 24px;**

**color: #ff0000;**

**margin-bottom: 20px;**

**}**

**#buyNow {**

**padding: 10px 20px;**

**font-size: 18px;**

**background-color: #0E4AA0;**

**color: white;**

**border: none;**

**border-radius: 5px;**

**cursor: pointer;**

**}**

**#buyNow:disabled {**

**background-color: #999;**

**cursor: not-allowed;**

**}**

**</style>**

**</head>**

**<body>**

**<h2>Limited Time Deal!</h2>**

**<div id="timer">05:00</div>**

**<button id="buyNow">Buy Now</button>**

**<script>**

**$(document).ready(function () {**

**let minutes = 5;**

**let seconds = 0;**

**function updateTimerDisplay() {**

**const min = String(minutes).padStart(2, '0');**

**const sec = String(seconds).padStart(2, '0');**

**$('#timer').text(`${min}:${sec}`);**

**}**

**const timerInterval = setInterval(function () {**

**if (seconds === 0) {**

**if (minutes === 0) {**

**clearInterval(timerInterval);**

**$('#buyNow').prop('disabled', true);**

**$('#timer').text("00:00");**

**return;**

**} else {**

**minutes--;**

**seconds = 59;**

**}**

**} else {**

**seconds--;**

**}**

**updateTimerDisplay();**

**}, 1000);**

**updateTimerDisplay();**

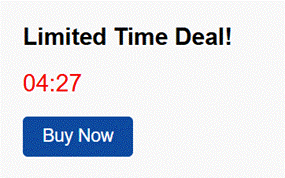
**});**

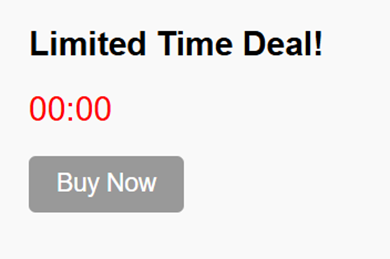
**</script>**

**</body>**

**</html>**

**Output :**

****

****

****

**Implementation :**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>Real-time Blog Search</title>**

**<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>**

**<style>**

**body {**

**font-family: Arial, sans-serif;**

**padding: 30px;**

**}**

**#search {**

**padding: 10px;**

**font-size: 18px;**

**width: 300px;**

**margin-bottom: 20px;**

**}**

**.article {**

**margin-bottom: 10px;**

**font-size: 18px;**

**}**

**mark {**

**background-color: yellow;**

**font-weight: bold;**

**}**

**</style>**

**</head>**

**<body>**

**<h2>Search Blog Articles</h2>**

**<input type="text" id="search" placeholder="Type to search...">**

**<div id="articles">**

**<div class="article">How to Learn JavaScript Effectively</div>**

**<div class="article">Understanding CSS Flexbox and Grid</div>**

**<div class="article">10 Tips for Productive Remote Work</div>**

**<div class="article">The Future of Artificial Intelligence</div>**

**<div class="article">Building Responsive Websites with Bootstrap</div>**

**</div>**

**<script>**

**$(document).ready(function () {**

**$('#search').on('input', function () {**

**let query = $(this).val().toLowerCase();**

**$('.article').each(function () {**

**let text = $(this).text();**

**let lowerText = text.toLowerCase();**

**if (query && lowerText.includes(query)) {**

**let regex = new RegExp(`(${query})`, 'gi');**

**let highlighted = text.replace(regex, '<mark>$1</mark>');**

**$(this).html(highlighted).show();**

**} else if (query === '') {**

**$(this).html(text).show();**

**} else {**

**$(this).hide();**

**}**

**});**

**});**

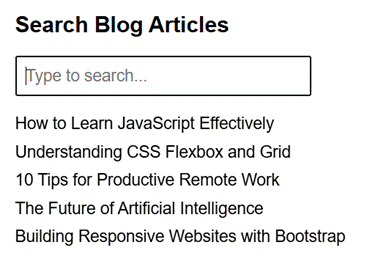
**});**

**</script>**

**</body>**

**</html>**

**Output :**

****

****