MIT Art, Design and Technology University
MIT School of Computing, Pune

Department of Information Technology

Lab Manual

Practical - Web Programming

Class - S.Y. (SEM-II), DA

Batch - DA-I/II

ASAD JIWANI

Mr./Ms.

A.Y. 2024 – 2025 (SEM-II)

File Index page given in the stationary

Web Programming

SEMESTER – IV

Course Code:	23IT2008	**Course Credits:**	02
Teaching Hours / Week (L:T:P):	0:0:4	**CA Marks:**	25
Total Number of Teaching Hours:		**END-SEM Marks:**	25

Course Pre-requisites:

Course Description:

This course provides a comprehensive introduction to web technology, designed to help students develop a strong foundation in building and managing websites and web applications. The curriculum covers key topics such as HTML, CSS, and JavaScript,PHP, MySQL, which are essential for creating interactive, well-designed web pages. Students will also explore the principles of responsive design, ensuring that web applications are optimized for different devices and screen sizes.

The course dives deeper into server-side technologies, including HTTP, web servers, and databases, allowing students to understand how websites function behind the scenes. Emphasis is placed on practical learning, and students will gain hands-on experience by working on projects that showcase their ability to design, develop, and deploy websites.

By the end of the course, students will be proficient in using modern web technologies to create web applications. They will understand how to handle client-server interactions, manage user data, and implement various web technologies to enhance the functionality of their applications.

Course Learning Objectives: This course will enable the students to:

1. Understand fundamental concepts of front-end web development.
2. Enable students to create basic web pages incorporating essential elements such as images, hyperlinks, lists, tables, and forms.
3. Teach students how to use CSS to manage fonts, lists, colors, text alignment, and background images for a cohesive and aesthetically pleasing web design.
4. Develop an understanding of JavaScript scopes to manage the visibility and lifetime of variables and functions effectively.
5. Equip students with the skills to implement and handle JavaScript events, enabling enhanced user interactions through event-driven programming.
6. Apply comprehensive knowledge of HTML, CSS, and JavaScript to develop a complete front-end application. Utilize project-based learning to showcase problem-solving skills and creativity in web development projects.
7. Configure server environments with Apache/TOMCAT.
8. Set up a PHP development environment and write basic PHP scripts.
9. Master PHP programming constructs for web development tasks.
10. Create and process HTML forms, and manage MySQL database operations.
11. Develop comprehensive back-end applications using PHP and MySQL.

Course Outcome: After taking this course, Students will be able to :

1. Apply knowledge of HTML to create the structure of the webpage and CSS to style and layout the elements, making the application visually appealing.
2. Apply comprehensive knowledge of HTML, CSS, and JavaScript to develop a complete front-end application and utilize project-based learning to showcase problem-solving skills and creativity in web development projects.
3. Set up and configure a server environment using tools like Apache or TOMCAT and set up a PHP development environment. Write & execute simple PHP scripts, understanding PHP syntax and basic features, create HTML forms to collect user data and integrate with PHP for processing.
4. Design and develop a back-end application using PHP and MySQL, implementing CRUD

operations to manage data effectively.

UNIT – I	Introduction to HTML and Cascading Style Sheet	09 Hours

Module 1 - Markup Language (HTML): Introduction to HTML, Formatting and Fonts, Commenting Code, Anchors, Backgrounds, Images, Hyperlinks, Lists, Tables, Frames, HTML Forms
Module 2 - CSS: Need for CSS, introduction to CSS, basic syntax and structure, Levels of style sheets, Style specification formats, BOX Model, Selector forms, Property value forms, Font properties, List properties, Color, Alignment of text, Background images

Pedagogy	**ICT Teaching / PowerPoint Presentation and Videos:** **Use tools like Visual Studio Code (free).** **Videos:** **https://www.coursera.org/learn/html-css-javascript-for-web-developers**
	Self-study / Do it yourself /: **Practice creating basic HTML pages and enhancing them using CSS.**
	Experiential Learning Topics: **Design a simple webpage for coffee shop website**
	PBL - Project Based Learning: **Create a multi-page website (e.g., coffee shop website) using HTML and CSS.**

UNIT – II	Front-End Development	09 Hours

Module 3 - Overview of JavaScript, including JS in an HTML (Embedded, External), Basic JS syntax, basic interaction with HTML
Module 4 - Core features of JavaScript: Data types, Control Structures, Arrays, Functions and Scopes

Pedagogy	**ICT Teaching / PowerPoint Presentation and Videos:** **Use tools like Visual Studio Code (free).** **Videos:** **https://www.coursera.org/learn/javascript-basics**
	Self-study / Do it yourself /: **Solve exercises on JavaScript syntax, control structures, and functions**
	Experiential Learning Topics: **Build a web page with interactive elements (e.g., a simple calculator).**
	PBL - Project Based Learning: **Develop an interactive webpage that uses JavaScript to validate form inputs or perform basic calculations.**

UNIT – III	Advanced Front-End Development	09 Hours

Module 5 - DOM: DOM levels, DOM Objects and their properties and methods, Manipulating DOM
Module 6 - JavaScript Events: JavaScript Events, Types of JavaScript Events, Objects in JS, Event Handling

Pedagogy	**ICT Teaching / PowerPoint Presentation and Videos:** **https://www.coursera.org/learn/building-interactive-web-pages-using-**

	javascript **Use tools like Visual Studio Code (free).**
	Self-study / Do it yourself /: **Practice exercises on DOM traversal and event handling.**
	Experiential Learning Topics: **Add dynamic behavior to a webpage using DOM and events (e.g., a to-do list app).**
	PBL - Project Based Learning: **Develop a web page with dynamic content (e.g., a task manager or interactive quiz) using DOM manipulation and event handling.**

UNIT – IV	**Server Side Scripting**	**09 Hours**

Module 7 - Set up and configure a server environment using tools like Apache or TOMCAT, set up a PHP development environment.
Module 8 -Introduction to PHP: : Introduction to PHP, Server side scripting Vs Client side scripting, Basic Development Concepts (Mixing PHP with HTML), Creating, Writing & Running First PHP Script, PHP syntax, conditions & Loops, Functions, String manipulation, Arrays & Functions,
Module 9 - Form handling with HTML and PHP: Designing of Forms using HTML, Form Handling using GET and POST methods of Form

Pedagogy	**ICT Teaching / PowerPoint Presentation and Videos:** **https://www.coursera.org/learn/web-applications-php** **Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management**
	Self-study / Do it yourself /: **Practice exercises on form handling and server-side scripting with PHP.**
	Experiential Learning Topics: **Create a basic form for data submission and handle it using PHP (e.g., feedback form).**
	PBL - Project Based Learning: **Develop a small server-side application (e.g., a contact form with email validation and submission).**

UNIT – V	**Working with Databases and Web Application Development**	**09 Hours**

Module 10 - Working with databases using MySQL with PHP: MySQL database, create database, create table, primary key with AUTO_INCREMENT setting, Insert Data Into a Database Table, Select Data From a Database Table, Open or close a Connection to the MySQL Server.

Module 11 - Web Application Development (Project): Develop the web application to handle client-server interactions, manage user data, and implement various web technologies to enhance the functionality of their applications. Example: Website for a Coffee Shop

Pedagogy	**ICT Teaching / PowerPoint Presentation and Videos:** **Use tools like Visual Studio Code (free), XAMPP/WAMP for PHP server setup, and MySQL Workbench for database management** **Videos:** **https://www.coursera.org/learn/web-app**
	Self-study / Do it yourself /:

	Exercises on creating and manipulating databases using PHP and MySQL.
	Experiential Learning Topics: **Create a database and design a webpage to display its data dynamically.**
	PBL - Project Based Learning: **Develop a fully functional web application (e.g., a Coffee Shop website or e-commerce platform) that integrates database functionality for data management.**

Experiment No.1

Problem Statement:

1. Create the basic structure of the second-hand gaming console store website, including the home page layout with a header, main content area, and footer.

Prepare a common project website design and plan document for all assignments. Consider following points:

1. Brief information about the project.
2. Set the goals & deliverables.
3. Finalize the modules of the project.
4. Define the audience.
5. Describe pain points & the ideal experience (On the basis of existing systems)
6. Set the visual direction
7. Map out the Project structure.
8. Plan the content for each page.
9. Add ideas for content, images & layout.
10. Determine your site structure or Create content for your core website pages:

a. Home page

b. About page

c. Product/Service page

d. Testimonial/review page

e. Support page

f. Starter blog posts

11. Create and collect design elements

12. These design elements define your brand personality and help customers feel what your brand represents through the use of:

a. Colors

b. Fonts and typography

c. Logos

d. Images and photos

Objective:

To design the basic structure of a second-hand gaming console store website by planning its layout, content, and visual elements, ensuring it meets user needs and effectively represents the brand.

Theory:

1. Brief Information about the Project

The project aims to create a user-friendly and visually appealing website for "Huice Wrld," a second-hand gaming console store. The website will target console enthusiasts and customers interested in affordable gaming options. Key features include showcasing products, customer testimonials, and providing a secure login and registration system to personalize the user experience. The website will also include an online purchase option and mobile responsiveness to improve accessibility.

2. Goals and Deliverables

Goals:

-

Develop a functional and engaging website for Huice Wrld.

-
-

Showcase the store's products, customer reviews, and contact details.

-
-

Enable users to register, log in, and personalize their experience.

-
-

Create a responsive website that adapts to mobile, tablet, and desktop devices.

-

Deliverables:

-

Website Pages:

-
  -

Home Page

-
-

About Page

-
-

Products/Services Page

-
-

Testimonials Page

-
-

Contact Page

-
-

Login Page

-
-

Registration Page

-
-

Optional blog posts or placeholder for future blogs

-
-

Core Features:

-
-

Consistent header and footer navigation.

-
-

Functional login and registration system.

-
-

Mobile-responsive design.

-
-

Professional, user-friendly design with appropriate use of colors, fonts, and images.

-

3. Finalize the Modules of the Project

Website Modules:

1.

Home Page Module:

2.

-

Hero section with a call-to-action (e.g., "Explore Now").

-
-

Overview of featured products and promotions.

-
-

Navigation to other sections like About, Products, Testimonials, and Contact.

-
-

Footer with contact details and social media links.

-

3.

About Page Module:

4.

-

Introduction to Huice Wrld, its mission, and values.

-
-

Showcase the store's history and the quality of second-hand gaming consoles.

-

5.

Products/Services Page Module:

6.

-

A categorized catalog for different gaming consoles.

-
-

Product images, descriptions, and prices.

-
-

A search/filter option for users.

-

7.

Testimonials Page Module:

8.

-

A layout showcasing positive customer reviews.

-
-

Option for customers to submit their own reviews.

-

9.

Contact Page Module:

10.

- A contact form for user inquiries.
- A map of the physical store's location and contact details.

11.

Login Page Module:

12.

- User authentication for returning customers.
- Link to the registration page for new users.

13.

Registration Page Module:

14.

- A form to create an account, including Name, Email, and Password.
- Terms and conditions checkbox.

15.

Footer Module:

16.

- Footer with Privacy Policy, Terms of Service, and social media links.

4. Define the Audience

Target Audience:
The website is designed to cater to console enthusiasts, remote workers, students, health-conscious individuals, tourists, new users, and online shoppers.

- **Console Enthusiasts:** Detailed product descriptions, gaming tips, and exclusive offers.
- **Remote Workers:** Clear navigation to purchase consoles, in-store pickup, and contact page with hours.
- **Students:** Discounts and group offers prominently featured.
- **Health-Conscious Customers:** Catalog with health-related product filters (e.g., low-calorie, vegan).
- **Tourists:** Location details and regional gaming specialties.
- **New Users:** Clean interface, easy navigation, and clear store story.
- **Online Shoppers:** Secure e-commerce system with product categories.

-

5. Describe Pain Points & Ideal Experience

Pain Points:

-

Poor navigation and cluttered interfaces.

-
-

Limited online purchasing functionality.

-
-

Non-mobile optimized websites.

-
-

Insufficient product information.

-
-

Weak engagement strategies.

Inefficient contact and location details.

Lack of personalization.

Ideal Experience:

Intuitive Navigation: Sticky navigation bar with clear access to key pages.

Seamless Purchasing: Easy e-commerce system for browsing and completing purchases.

Mobile Responsiveness: Optimized for mobile-first design

Comprehensive Product Information: Clear, detailed product descriptions.

Engagement Features: Loyalty programs, blog, customer testimonials.

Easy Contact & Location Access: Simple forms, location map, and contact details.

Personalization: Account creation for saving preferences and receiving offers.

6. Set the Visual Direction

Visual Design Goals:

Welcoming and Comfortable: Cozy, approachable feel that reflects Huice Wrld's unique atmosphere.

-
-

Modern and Minimalistic: Clean, professional design with intuitive user experience.

-
-

Brand Representation: Colors, fonts, and images representing the essence of second-hand gaming consoles.

-

Core Visual Elements:

-

Color Palette:

-
    -

Console Brown (#6F4E37): Header, footer, buttons, and highlights.

    -
    -

Creamy Beige (#F5F5DC): Background for warmth and contrast.

    -
    -

Deep Espresso (#3C2F2F): Text and accents.

    -
    -

Olive Green (#556B2F): Call-to-action buttons.

-
-

Typography:

-
-

Primary: Poppins or Roboto (for headings and CTA).

-
-

Secondary: Open Sans or Lora (for body text).

-
-

Imagery & Icons:

-
-

High-quality images of gaming consoles and cozy store spaces.

-
-

Simple, minimalistic icons for navigation.

-
-

Hero Banner: Featured products or promotions.

-

1.

7. Map out the Project structure

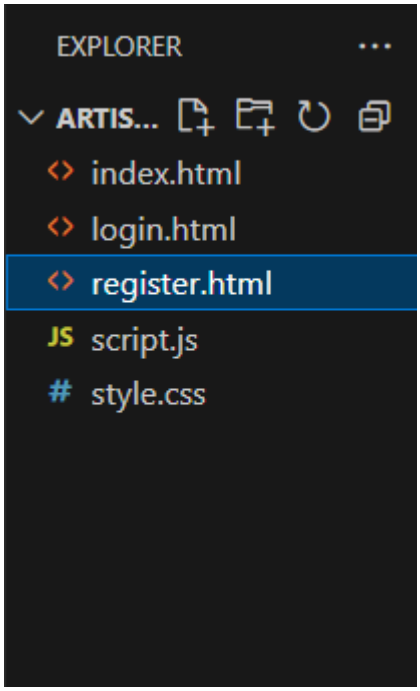
huice_wrld_website/

├── index.html # Home page

├── about.html # About page

├── products.html # Products/Services page

```
├── testimonials.html       # Testimonials page
├── contact.html            # Contact page
├── login.html              # Login page
├── register.html           # Registration page
├── blog.html               # Blog page (optional)
├── assets/
│   ├── css/
│   │   ├── style.css       # Global CSS
│   │   ├── responsive.css  # Mobile optimization
│   ├── js/
│   │   ├── main.js         # Scripts for interactivity
│   ├── images/
│   │   ├── logo.png        # Logo
│   │   ├── console_catalog/ # Product images
├── fonts/
│   ├── Poppins/            # Primary font
│   ├── OpenSans/           # Secondary font
└── README.md               # Documentation
```



8. Plan the content for each page

The website will include a minimum of 5 core pages, along with additional Login and Registration pages. This plan details the content for each page.

1. Home Page

Welcome fans and showcase featured Juice WRLD albums, merch, and exclusive drops.

Content Plan:

Header:

- Logo: Stylized "999" or Juice WRLD branding.
- Navigation: Home, About, Albums, Testimonials, Contact.
- CTA: Login / Sign Up.

Hero Section:

- Full-width background: Juice WRLD concert photo or album cover.
- Tagline: “Legends Never Die. Keep His Music Alive.”
- CTA: “Explore Albums” / “Shop Merch”

About Preview:

- Brief intro: “Celebrating the life and legacy of Juice WRLD through exclusive albums and merchandise.”
- CTA: “Learn More About Juice”

Featured Albums Section:

- Carousel or grid of top Juice WRLD albums.
- Text: “Limited Vinyl Editions Available Now”

Footer:

- Quick Links, Social Media, Newsletter Signup, Contact Info.

2. About Page

Purpose:

Tell the story of Juice WRLD and the mission behind the album store.

Content Plan:

About Juice WRLD:

- Short biography, musical journey, legacy.
- Quote or lyrics highlight.

Mission Statement:

- "Our mission is to preserve and promote Juice WRLD's legacy through high-quality music, memorabilia, and fan engagement."

Team / Collaborators Section:

- Short bios of the team running the store or producers involved.

Special Features:

- "Why Shop With Us?": Authentic merch, exclusive content, fast shipping.

Footer:

- Same as Home Page.

3. Albums / Products Page

Purpose:
Showcase albums, merchandise, and collectibles.

Content Plan:

Categories:

- Albums (CD/Vinyl/Digital), Apparel, Posters, Accessories.

Product Grid:

- Product image, name, brief description, price.
- CTA: "Add to Cart" / "Buy Now"

Top Picks Section:

- Customer favorites like Goodbye & Good Riddance, Legends Never Die, Fighting Demons.

CTA Section:

- "Join the 999 Club for Exclusive Drops"

Footer:

- Same as Home Page.

4. Testimonials Page

Purpose:
Show fan reviews, ratings, and shared experiences.

Content Plan:

Fan Feedback:

- Carousel of testimonials with star ratings.

Submit Your Testimonial:

- Form to submit reviews, with fields for name, rating, comment.

Featured Reviews:

- Pull top reviews from social platforms or fan forums.

Footer:

- Same as Home Page.

5. Contact Page

Purpose:
Allow fans to get in touch.

Content Plan:

Contact Form:

- Name, Email, Subject, Message.
- Submit button.

Location / Virtual Address:

- Mailing address for business correspondence.

Operating Info:

- Business hours or response time estimates.

Social Media Section:

- Links to Instagram, X (Twitter), YouTube.

Footer:

- Same as Home Page.

6. Login Page

Content Plan:

Form Fields:

- Email and Password.
- Submit button.

Forgot Password Link:

Sign-Up Prompt:

- "Don't have an account? Sign Up Now!"

7. Registration Page

Content Plan:

Form Fields:

- Full Name, Email, Password, Confirm Password.

CTA:

- "Create Account"

Newsletter Checkbox:

8. Starter Blog Posts

Categories:

- Album Reviews: Deep dives into Juice WRLD's discography.
- Legacy: How his music continues to impact fans.
- Behind the Music: Studio insights, collaborators.
- Mental Health & Lyrics: Analysis and awareness.

9. Visual Design Elements

Colors:

- Primary: Deep Purple (#5E3A87), Midnight Black (#1A1A1A), Cloud White (#F3F3F3)
- Accent: 999 Red (#FF3C38), Neon Blue (#3CCEFF)

Typography:

- Headings: Lora or Bebas Neue (bold, impactful)
- Body: Roboto or Open Sans (modern, clean)

Logo:

- A minimalist "999" mark or Juice WRLD silhouette.

Images:

- Album covers, fan art, vinyl setups, concert shots.

Buttons & Interactive Elements:

- H
- over effects in red/blue tones.
- Icons for cart, account, heart/favorite, and music notes.

10. Determine your site structure or Create content for your core website pages:

a. Home page

Header:

-

Logo: A stylized Juice WRLD logo (e.g., a combination of his signature and a musical element).

-
-

Navigation Links: Home, About, Albums, Testimonials, Contact.

-
-

Call-to-Action Button: "Explore Juice WRLD’s Albums" (links to the products page).

-

Hero Section:

-

Background Image: Full-width image of Juice WRLD performing or album covers.

-
-

Text Overlay: "Live Forever with Juice WRLD's Albums".

-
-

Call-to-Action Button: "Shop Albums" (links to product pages).

-

About Section (Teaser):

-

Introduction: Brief paragraph introducing Juice WRLD and the store’s mission to provide his albums and legacy.

-
-

Link to About Page.

-

Featured Products Section:

-

Featured Albums: Showcasing 3-4 key albums, such as "Legends Never Die" and "Goodbye & Good Riddance."

-
-

Album Art, Short Descriptions, Price, "Buy Now" Button.

-

Social Proof Section (Testimonial Teaser):

-

Snippets from customer reviews praising the store's service.

-
-

“See More” Button linking to full testimonials.

-

Footer:

-

Quick Links: Catalog, Store Hours, Locations, FAQs.

-
-

Social Media Icons linking to Juice WRLD's Instagram, Twitter, and YouTube.

-
-

Store Address with Google Maps embed.

-

---

2. About Page

Introduction:

-

Overview of Juice WRLD, his impact on the music world, and the store's mission to keep his legacy alive by sharing his music with the world.

-
-

Brief history: "Founded in 2025, our mission is to provide fans access to the best of Juice WRLD's music while keeping his spirit alive."

-

Meet the Team:

-

Grid layout showcasing key team members with their names, photos, and brief bios (focus on those behind the album distribution and customer service).

-

Our Promise:

-

Information on how albums are sourced, packaged, and distributed with care.

-
-

Eco-friendly practices: Using recycled materials for packaging.

-

Location Section:

-

List of physical store locations (if any) with Google Maps integration.

-
-

Hours of operation.

-

---

3. Albums Page

Album Categories:

-

Categories: Albums, Vinyl, Merchandise, Special Editions.

-
-

Each product should include an image, description, features, and price.

-
-

"Add to Cart" or "Buy Now" button.

-

Popular Items & Limited-Time Specials:

-

Featured box or carousel showcasing special offers on popular albums or exclusive merchandise.

-

Footer (same as Home Page):

-

Quick links, social media icons, store locations.

-

---

4. Testimonials/Review Page

Customer Reviews:

-

Carousel or grid layout of reviews, each with a star rating, testimonial, and customer name.

-
-

Submit Your Review button for customers to leave feedback.

-

Featured Reviews:

-

Pull reviews from social media, like Instagram or Twitter, to build trust.

-

Reviewing Process Section:

-

A brief explanation of how reviews are managed and shared.

-

---

5. Contact Page

Contact Form:

-

Fields: Name, Email, Message.

-
-

Submit button.

-

Social Media & Address Section:

-

Icons linking to Juice WRLD's Instagram, Twitter, YouTube, etc.

-
-

Full address, phone number, email for inquiries.

-

Interactive Map:

-

Google Maps integration for store locations.

-

Support Information:

-

Customer support contact details and FAQs.

-

---

6. Blog Section

Blog Categories:

-

Juice WRLD's Legacy: Articles discussing his influence on music and culture.

-
-

Behind the Scenes: Features on the creation of Juice WRLD's albums, interviews with producers, etc.

-
-

Merchandise Drops: Updates on new Juice WRLD merch or album releases.

-
-

Community Engagement: Stories about fan involvement and charity events.

-

---

7. Login & Registration Pages

Login Page:

-

Fields: Username/email and password.

-
-

Forgot password? link.

-

Registration Page:

-

Fields: Name, Email, Password, and an option to subscribe to newsletters or special offers.

-

Design Elements for the Juice WRLD Store

Colors:

-

Primary Colors: Black (#111), White (#fff), Red (#FF0000), Dark Purple (#6A0DAD).

-
-

Accent Colors: Gold (#FFD700), Silver (#C0C0C0) for a more premium feel.

-

Typography:

-

Heading Font: Playfair Display or Lora (serif), reflecting the elegance of Juice WRLD's impact.

-
-

Body Font: Open Sans or Roboto (sans-serif), clean and easy to read.

-
-

Font Weights: Bold for headings, regular for text.

-

Logo:

-

Logo Design: A sleek, modern logo combining Juice WRLD's signature with a subtle music symbol (e.g., a headphone or microphone).

-
-

Color Palette for Logo: Black, Purple, and Red tones.

-

Imagery and Photos:

-

High-quality Album Art: Displaying Juice WRLD's album covers.

-
-

Lifestyle Photography: Fans enjoying his music, merchandise, and events.

-

Interactive Elements:

-

Buttons: Rich red or gold for CTA buttons ("Buy Now", "Subscribe").

-
-

Icons: Clean, simple icons that match the branding (e.g., headphone icon for music, cart for purchases).

-

Conclusion:

The **Juice WRLD Albums Store** website design aims to offer fans of the late rapper a seamless and engaging platform where they can explore, purchase, and celebrate his musical legacy. Through careful planning and the use of intuitive design elements, we have created a layout that captures Juice WRLD's essence while providing a user-friendly experience.

By organizing the website into clear, well-defined sections—such as the Home, About, Albums, Testimonials, Contact, and Blog pages—the site ensures easy navigation and access to the essential

content. The product pages highlight his music, albums, and related merchandise, with calls to action designed to encourage fans to engage with the content and make purchases.

The design elements, including the color palette, typography, and imagery, reflect Juice WRLD's unique artistic style and cultural impact. High-quality images, vibrant colors, and interactive features will ensure the website feels both authentic to his brand and inviting to users.

Through the inclusion of user reviews, a blog section, and interactive elements, the site fosters a sense of community and engagement, allowing fans to connect with each other and with the legacy of Juice WRLD.

Ultimately, the **Juice WRLD Albums Store** website will not only serve as a platform for purchasing music and merchandise but also as a tribute to the artist's impact on the music world and his dedicated fan base.

Experiment No.2

Problem Statement:

- Create a detailed home page for the coffee shop website.
- Create a detailed menu/product page for the coffee shop website, listing all available items categorized appropriately.
- Create a cart page that allows customers to review and manage the items they wish to purchase before proceeding to checkout.
- Create an about us page that provides detailed information about the coffee shop's history, mission, and team.
- Create a contact page that allows customers to easily get in touch with the coffee shop through a form.
- Design and implement admin/user registration form for the coffee shop website.
- Design and implement admin/user login form for the coffee shop website.

Objective:

To create a Juice WRLD Album Store webpage using HTML

Introduction

In today's digital era, e-commerce platforms are essential for offering music to fans worldwide. This project focuses on creating a responsive and functional website for a Juice WRLD album store. The platform caters to fans looking for Juice WRLD's iconic albums, vinyl, and merchandise, promoting a deeper connection to his music and legacy.

The website integrates front-end components like album listings, user authentication, cart management, and contact forms, all using HTML, CSS, and potentially JavaScript or server-side scripting in later phases.

---

1. Home Page

The home page serves as the landing page and provides a snapshot of Juice WRLD's offerings. It typically includes:

-

A hero section with album promotions or best-sellers.

-
-

A navigation bar for easy access to other sections.

-
-

Call-to-action buttons ("Shop Now," "Explore Juice WRLD’s Music," etc.)

-
-

Customer testimonials or featured albums.

-

Importance:
The home page establishes first impressions and reinforces the artist’s brand. A clean, visually appealing layout will keep users engaged and reduce bounce rates.

Technologies used:
HTML for structure, CSS for layout and visuals, optional animations using CSS or JavaScript for interactivity.

---

2. Album/Menu Page

This page displays Juice WRLD’s entire catalog of albums, vinyl, and merchandise. Items are grouped into categories such as:

-

Albums (e.g., "Legends Never Die," "Goodbye & Good Riddance")

-
-

Vinyl

-
-

Merchandise (T-shirts, posters, etc.)

-
-

Special Editions

-

Each product should include:

-

Album cover image

-
-

Title and specifications

-
-

Price

-
-

“Add to Cart” button

-

Importance:
A structured catalog improves discoverability and allows fans to compare and select the albums and products they want.

UX Consideration:
Filters by album type (e.g., vinyl, digital, merch) improve usability and customer satisfaction.

---

3. Cart Page

The cart page displays:

-

All added items with quantity and subtotal.

-
-

Options to update or remove items.

-
-

A **final checkout button**.

-

Real-world relevance:
This page gives fans control over their purchases before checkout.

Optional enhancements:

-

Cart persistence using localStorage.

-
-

Live price updates when quantity is changed.

-

---

4. About Us Page

This section gives the business a personal touch, including:

-

History of Juice WRLD and the store's mission to provide his music to fans globally.

-
-

Message from the founder or store team.

-
-

Team photos and bios.

-

Purpose:
Builds trust and authenticity with potential buyers, especially when dealing with beloved artists' legacy.

---

5. Contact Page

A contact form is essential for customer support, with:

- **Name**
- **Email**
- **Message**

Additional elements:

- **Phone number and address**
- **Map location using Google Maps**
- **Social media links**

UX Factor:
Quick communication fosters better customer support and enhances customer satisfaction.

---

6. User/Admin Registration Form

This page allows new users or admins to create an account. Fields include:

- **Full name**

-
-

Email or phone

-
-

Password and confirmation

-
-

User type (dropdown or radio buttons)

-

Functionality:

-

Form validation (password match, email format)

-
-

Secure data storage (back-end database for real deployment)

-

Why it matters:
Allowing personalized experiences for users and secure access for admins ensures a seamless platform experience.

---

7. User/Admin Login Form

This form validates users or admins and redirects them to respective dashboards.
Fields include:

-

Username/email

-
-

Password

-

-

Remember me checkbox

-
-

Forgot password link

-

Security Considerations:

-

Basic input validation

-
-

In production: hashing passwords, rate limiting, and two-factor authentication.

-

Differentiated Access:

-

Users can shop, view order history.

-
-

Admins can manage inventory, view analytics, and process orders.

-

Technological Stack Overview (Future Enhancement)

While this version is made using HTML/CSS, it can be extended later with:

-

JavaScript for dynamic features (live cart updates, animations).

-
-

PHP/Node.js for server-side logic.

-
-

MySQL/MongoDB for database storage.

-
-

Session management and authentication for secure login systems.

-

Sustainability Impact

The store promotes eco-conscious consumerism by extending the life cycle of albums and merchandise. It reduces waste and supports circular economy practices by:

-

Offering **vinyl** and **exclusive editions** to preserve Juice WRLD's legacy.

-
-

Educating fans about the significance of preserving music and culture through his albums.

-

Code:

A. Home page:

code:<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Juice WRLD Albums</title>

<link rel="stylesheet" href="styles.css">

<style>

body {

```
    font-family: 'Arial', sans-serif;
    text-align: center;
    background: linear-gradient(to right, #1a1a1a, #333);
    color: white;
    margin: 0;
    padding: 0;
}
header {
    background: #111;
    color: white;
    padding: 20px 0;
    font-size: 24px;
    font-weight: bold;
    text-transform: uppercase;
    font-family: 'Courier New', Courier, monospace;
}
.nav-bar {
    display: flex;
    justify-content: center;
    gap: 20px;
    background: #222;
    padding: 10px;
}
.nav-bar a {
    color: #ff6600;
    text-decoration: none;
    font-size: 16px;
    transition: color 0.3s;
}
.nav-bar a:hover {
```

```
    color: #fff;
}
.album-container {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    gap: 20px;
    margin-top: 20px;
}
.album {
    background: #222;
    padding: 15px;
    border-radius: 10px;
    box-shadow: 0px 4px 8px rgba(255, 102, 0, 0.3);
    width: 250px;
    transition: transform 0.3s;
}
.album:hover {
    transform: scale(1.05);
}
.album img {
    width: 100%;
    border-radius: 10px;
}
.add-to-cart {
    background: #ff6600;
    color: white;
    padding: 10px 15px;
    border: none;
    border-radius: 5px;
```

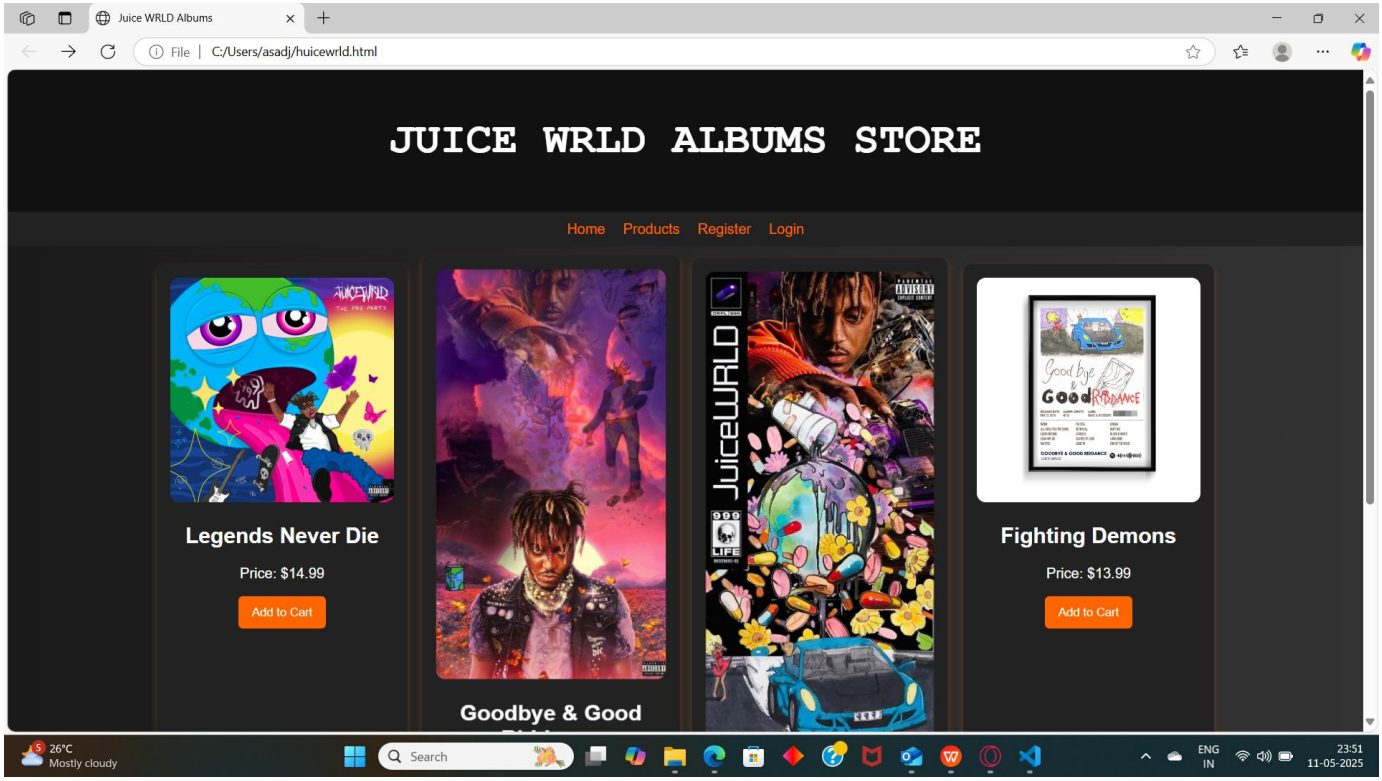
```
            cursor: pointer;
        }
        .cart-container {
            margin-top: 30px;
            padding: 20px;
            background: #222;
            border-radius: 10px;
        }
    </style>
</head>
<body>
    <header>
        <h1>Juice WRLD Albums Store</h1>
    </header>
    <nav class="nav-bar">
        <a href="index.html">Home</a>
        <a href="products.html">Products</a>
        <a href="register.html">Register</a>
        <a href="login.html">Login</a>
    </nav>
    <div class="album-container">
        <div class="album">
            <img src="https://i.scdn.co/image/ab67616d0000b273ed1cc06f375a38c942c0c3e5">
            <h2>Legends Never Die</h2>
            <p>Price: $14.99</p>
            <button class="add-to-cart" onclick="addToCart('Legends Never Die', 14.99)">Add to Cart</button>
        </div>
        <div class="album">
            <img src="https://i.redd.it/92p7e1pehrx51.jpg">
            <h2>Goodbye & Good Riddance</h2>
```

```
        <p>Price: $12.99</p>
        <button class="add-to-cart" onclick="addToCart('Goodbye & Good Riddance', 12.99)">Add to Cart</button>
    </div>
    <div class="album">
        <img src="https://i.pinimg.com/474x/46/3f/0e/463f0e4ca6804044d5bca227a467b455.jpg">
        <h2>Death Race for Love</h2>
        <p>Price: $15.99</p>
        <button class="add-to-cart" onclick="addToCart('Death Race for Love', 15.99)">Add to Cart</button>
    </div>
    <div class="album">
        <img src="https://themortalsoul.com/cdn/shop/files/44_e7cb768d-7188-4a1a-ba04-81ed7ede243a.png?v=1735449715">
        <h2>Fighting Demons</h2>
        <p>Price: $13.99</p>
        <button class="add-to-cart" onclick="addToCart('Fighting Demons', 13.99)">Add to Cart</button>
    </div>
</div>

<div class="cart-container">
    <h2>Shopping Cart</h2>
    <ul id="cart-items"></ul>
    <h3>Total: $<span id="total-price">0.00</span></h3>
</div>

<script>
    let cart = [];
    let total = 0;

    function addToCart(albumName, price) {
        cart.push({ name: albumName, price: price });
        total += price;
```

```
        updateCart();
    }

    function updateCart() {
        const cartList = document.getElementById('cart-items');
        cartList.innerHTML = '';
        cart.forEach((item, index) => {
            const li = document.createElement('li');
            li.textContent = `${item.name} - $${item.price.toFixed(2)}`;

            const removeBtn = document.createElement('button');
            removeBtn.textContent = 'Remove';
            removeBtn.onclick = () => removeFromCart(index);

            li.appendChild(removeBtn);
            cartList.appendChild(li);
        });
        document.getElementById('total-price').innerText = total.toFixed(2);
    }

    function removeFromCart(index) {
        total -= cart[index].price;
        cart.splice(index, 1);
        updateCart();
    }
  </script>
</body>
</html>
```

Output:

A. Index/Home page output:



Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Juice WRLD Albums</title>

  <link rel="stylesheet" href="styles.css">

  <style>

    body {

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

      text-align: center;
```

```
    background: linear-gradient(to right, #1a1a1a, #333);
    color: white;
    margin: 0;
    padding: 0;
}
header {
    background: #111;
    color: white;
    padding: 20px 0;
    font-size: 24px;
    font-weight: bold;
    text-transform: uppercase;
    font-family: 'Courier New', Courier, monospace;
}
.nav-bar {
    display: flex;
    justify-content: center;
    gap: 20px;
    background: #222;
    padding: 10px;
}
.nav-bar a {
    color: #ff6600;
    text-decoration: none;
    font-size: 16px;
    transition: color 0.3s;
}
.nav-bar a:hover {
    color: #fff;
```

```
}
.album-container {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    gap: 20px;
    margin-top: 20px;
}
.album {
    background: #222;
    padding: 15px;
    border-radius: 10px;
    box-shadow: 0px 4px 8px rgba(255, 102, 0, 0.3);
    width: 250px;
    transition: transform 0.3s;
}
.album:hover {
    transform: scale(1.05);
}
.album img {
    width: 100%;
    border-radius: 10px;
}
.add-to-cart {
    background: #ff6600;
    color: white;
    padding: 10px 15px;
    border: none;
    border-radius: 5px;
```

```
            cursor: pointer;
        }
        .cart-container {
            margin-top: 30px;
            padding: 20px;
            background: #222;
            border-radius: 10px;
        }
    </style>
</head>
<body>
    <header>
        <h1>Juice WRLD Albums Store</h1>
    </header>
    <nav class="nav-bar">
        <a href="index.html">Home</a>
        <a href="products.html">Products</a>
        <a href="register.html">Register</a>
        <a href="login.html">Login</a>
    </nav>
    <div class="album-container">
        <div class="album">
            <img src="https://i.scdn.co/image/ab67616d0000b273ed1cc06f375a38c942c0c3e5">
            <h2>Legends Never Die</h2>
            <p>Price: $14.99</p>
            <button class="add-to-cart" onclick="addToCart('Legends Never Die', 14.99)">Add to Cart</button>
        </div>
        <div class="album">
            <img src="https://i.redd.it/92p7e1pehrx51.jpg">
```

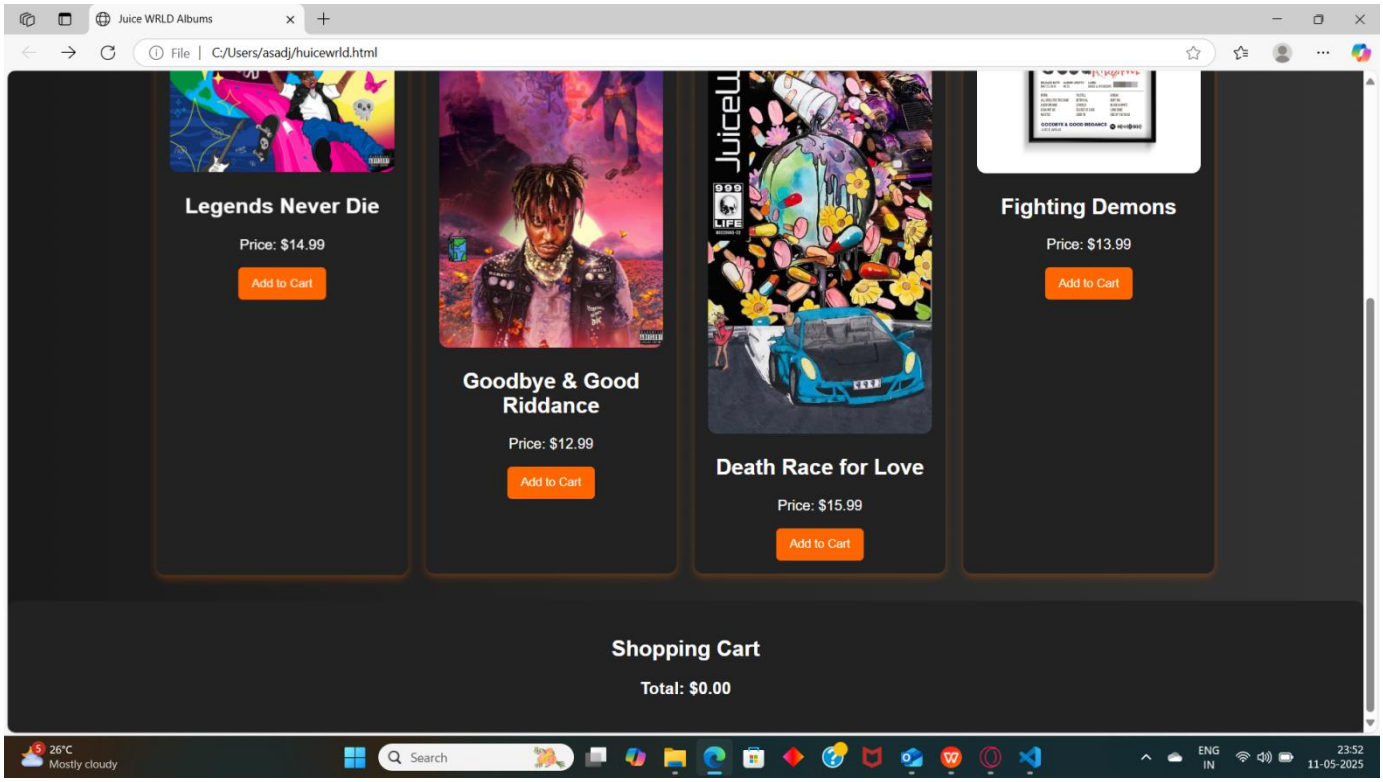
```
        <h2>Goodbye & Good Riddance</h2>
        <p>Price: $12.99</p>
        <button class="add-to-cart" onclick="addToCart('Goodbye & Good Riddance', 12.99)">Add to Cart</button>
    </div>
    <div class="album">
        <img src="https://i.pinimg.com/474x/46/3f/0e/463f0e4ca6804044d5bca227a467b455.jpg">
        <h2>Death Race for Love</h2>
        <p>Price: $15.99</p>
        <button class="add-to-cart" onclick="addToCart('Death Race for Love', 15.99)">Add to Cart</button>
    </div>
    <div class="album">
        <img src="https://themortalsoul.com/cdn/shop/files/44_e7cb768d-7188-4a1a-ba04-
81ed7ede243a.png?v=1735449715">
        <h2>Fighting Demons</h2>
        <p>Price: $13.99</p>
        <button class="add-to-cart" onclick="addToCart('Fighting Demons', 13.99)">Add to Cart</button>
    </div>
</div>

<div class="cart-container">
    <h2>Shopping Cart</h2>
    <ul id="cart-items"></ul>
    <h3>Total: $<span id="total-price">0.00</span></h3>
</div>

<script>
    let cart = [];
    let total = 0;
```

```
    function addToCart(albumName, price) {
        cart.push({ name: albumName, price: price });
        total += price;
        updateCart();
    }

    function updateCart() {
        const cartList = document.getElementById('cart-items');
        cartList.innerHTML = '';
        cart.forEach((item, index) => {
            const li = document.createElement('li');
            li.textContent = `${item.name} - $${item.price.toFixed(2)}`;

            const removeBtn = document.createElement('button');
            removeBtn.textContent = 'Remove';
            removeBtn.onclick = () => removeFromCart(index);

            li.appendChild(removeBtn);
            cartList.appendChild(li);
        });
        document.getElementById('total-price').innerText = total.toFixed(2);
    }

    function removeFromCart(index) {
        total -= cart[index].price;
        cart.splice(index, 1);
        updateCart();
    }
</script>
```

```
</body>

</html>
```

Output:

B. menu/product page output:



Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Juice WRLD Albums</title>

    <link rel="stylesheet" href="styles.css">

    <style>

        body {

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

            text-align: center;

            background: linear-gradient(to right, #1a1a1a, #333);
```

```
    color: white;
    margin: 0;
    padding: 0;
}
header {
    background: #111;
    color: white;
    padding: 20px 0;
    font-size: 24px;
    font-weight: bold;
    text-transform: uppercase;
    font-family: 'Courier New', Courier, monospace;
}
.nav-bar {
    display: flex;
    justify-content: center;
    gap: 20px;
    background: #222;
    padding: 10px;
}
.nav-bar a {
    color: #ff6600;
    text-decoration: none;
    font-size: 16px;
    transition: color 0.3s;
}
.nav-bar a:hover {
    color: #fff;
}
.album-container {
```

```
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    gap: 20px;
    margin-top: 20px;
}
.album {
    background: #222;
    padding: 15px;
    border-radius: 10px;
    box-shadow: 0px 4px 8px rgba(255, 102, 0, 0.3);
    width: 250px;
    transition: transform 0.3s;
}
.album:hover {
    transform: scale(1.05);
}
.album img {
    width: 100%;
    border-radius: 10px;
}
.add-to-cart {
    background: #ff6600;
    color: white;
    padding: 10px 15px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}
.cart-container {
```

```
            margin-top: 30px;
            padding: 20px;
            background: #222;
            border-radius: 10px;
        }
    </style>
</head>
<body>
    <header>
        <h1>Juice WRLD Albums Store</h1>
    </header>
    <nav class="nav-bar">
        <a href="index.html">Home</a>
        <a href="products.html">Products</a>
        <a href="register.html">Register</a>
        <a href="login.html">Login</a>
    </nav>

    <!-- Album Display Section -->
    <div class="album-container">
        <!-- Legends Never Die Album -->
        <div class="album">
            <img src="https://i.scdn.co/image/ab67616d0000b273ed1cc06f375a38c942c0c3e5" alt="Legends Never Die">
            <h2>Legends Never Die</h2>
            <p>Price: $14.99</p>
            <button class="add-to-cart" onclick="addToCart('Legends Never Die', 14.99)">Add to Cart</button>
        </div>

        <!-- Goodbye & Good Riddance Album -->
        <div class="album">
```

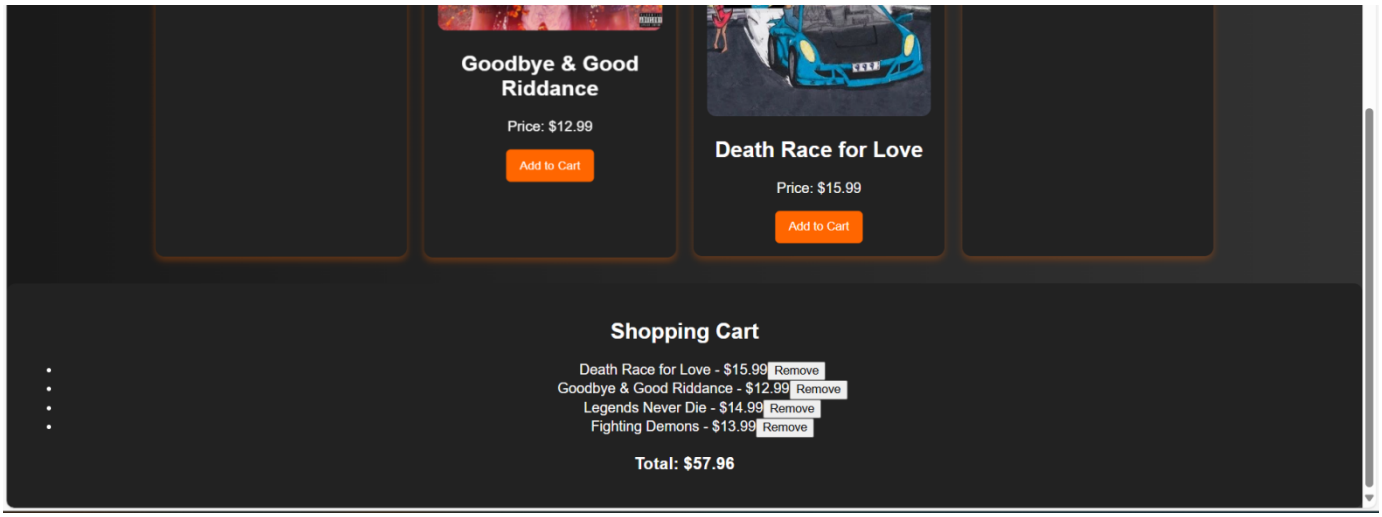
```
        <img src="https://i.redd.it/92p7e1pehrx51.jpg" alt="Goodbye & Good Riddance">
        <h2>Goodbye & Good Riddance</h2>
        <p>Price: $12.99</p>
        <button class="add-to-cart" onclick="addToCart('Goodbye & Good Riddance', 12.99)">Add to Cart</button>
    </div>

    <!-- Death Race for Love Album -->
    <div class="album">
        <img src="https://i.pinimg.com/474x/46/3f/0e/463f0e4ca6804044d5bca227a467b455.jpg" alt="Death Race for Love">
        <h2>Death Race for Love</h2>
        <p>Price: $15.99</p>
        <button class="add-to-cart" onclick="addToCart('Death Race for Love', 15.99)">Add to Cart</button>
    </div>

    <!-- Fighting Demons Album -->
    <div class="album">
        <img src="https://themortalsoul.com/cdn/shop/files/44_e7cb768d-7188-4a1a-ba04-81ed7ede243a.png?v=1735449715" alt="Fighting Demons">
        <h2>Fighting Demons</h2>
        <p>Price: $13.99</p>
        <button class="add-to-cart" onclick="addToCart('Fighting Demons', 13.99)">Add to Cart</button>
    </div>
</div>

<!-- Shopping Cart Section -->
<div class="cart-container">
    <h2>Shopping Cart</h2>
    <ul id="cart-items"></ul>
    <h3>Total: $<span id="total-price">0.00</span></h3>
</div>
```

```
<script>
    let cart = [];
    let total = 0;

    // Add item to cart
    function addToCart(albumName, price) {
        cart.push({ name: albumName, price: price });
        total += price;
        updateCart();
    }

    // Update cart display
    function updateCart() {
        const cartList = document.getElementById('cart-items');
        cartList.innerHTML = '';
        cart.forEach((item, index) => {
            const li = document.createElement('li');
            li.textContent = `${item.name} - $${item.price.toFixed(2)}`;

            // Remove item button
            const removeBtn = document.createElement('button');
            removeBtn.textContent = 'Remove';
            removeBtn.onclick = () => removeFromCart(index);

            li.appendChild(removeBtn);
            cartList.appendChild(li);
        });
        document.getElementById('total-price').innerText = total.toFixed(2);
    }
```

```
        // Remove item from cart
        function removeFromCart(index) {
            total -= cart[index].price;
            cart.splice(index, 1);
            updateCart();
        }
    </script>
</body>
</html>
```

Output:

C. cart page output:



Code:

About us page code -

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About Us - Juice WRLD Albums Store</title>
```

```
<style>
    body {
        font-family: Arial, sans-serif;
        background: #222;
        color: white;
        margin: 0;
        padding: 0;
    }
    header {
        background: #111;
        color: white;
        padding: 20px;
        text-align: center;
        font-size: 24px;
        font-weight: bold;
    }
    .nav-bar {
        text-align: center;
        background: #333;
        padding: 10px;
    }
    .nav-bar a {
        color: #ff6600;
        text-decoration: none;
        margin: 0 15px;
    }
    .about-container {
        max-width: 800px;
        margin: 30px auto;
        padding: 20px;
```

```
            background: #444;
            border-radius: 10px;
            text-align: center;
        }
        .about-container h2 {
            color: #ff6600;
        }
        .team {
            display: flex;
            justify-content: center;
            gap: 20px;
            margin-top: 20px;
        }
        .team-member {
            background: #333;
            padding: 15px;
            border-radius: 10px;
            text-align: center;
            width: 150px;
        }
        .team-member img {
            width: 100%;
            border-radius: 10px;
        }
    </style>
</head>
<body>
    <header>Juice WRLD Albums Store</header>
    <nav class="nav-bar">
        <a href="index.html">Home</a>
```

```
        <a href="products.html">Products</a>
        <a href="register.html">Register</a>
        <a href="login.html">Login</a>
        <a href="about.html">About Us</a>
    </nav>

    <div class="about-container">
        <h2>About Us</h2>
        <p>Welcome to the Juice WRLD Albums Store! We offer exclusive albums and memorabilia from the late Juice WRLD, providing fans with high-quality music and merchandise.</p>
        <p>Our mission is to keep Juice WRLD’s legacy alive through music, and we are here to offer a connection between the artist and his devoted fans.</p>

        <div class="team">
            <div class="team-member">
                <img src="https://via.placeholder.com/100" alt="Team Member">
                <p>John Doe - Founder</p>
            </div>
            <div class="team-member">
                <img src="https://via.placeholder.com/100" alt="Team Member">
                <p>Jane Smith - Manager</p>
            </div>
        </div>
    </div>
</body>
</html>
```

Output:

D. about us page output:

Code:

E. contact us page:

code:

```
<!DOCTYPE html>
<html>
<head>

  <title>Second hand consoles</title>
  <style>

    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

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

    nav {
      background-color: #333;
      padding: 1rem;
    }

    nav ul {
      display: flex;
      justify-content: center;
      list-style: none;
```

```
        }

        nav ul li {
            margin: 0 15px;
        }

        nav ul li a {
            color: white;
            text-decoration: none;
            font-weight: bold;
        }

        .content {
            background-color: white;
            padding: 20px;
            text-align: center;
            border-radius: 8px;
        }

    </style>
</head>
<body>

    <nav>
        <ul>
            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\homepage.html">Home</a></li>
            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\login.html">Login</a></li>
                                        <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\registration.html">Registration</a></li>
            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\contact_us.html">Contact Us</a></li>
            <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\about_us.html">About Us</a></li>
```

```
        <li><a href="C:\\Users\\ojas\\OneDrive\\Desktop\\WPL\\WPL TA-1\\product.html">Products</a></li>
    </ul>
</nav>

<div class="content">
    <h2>Contact Us</h2>

</div>
</body>
</html>
```

Code:

F. registration page:

code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: Arial, sans-serif;
        }

        body {
            background-color: #f4f4f4;
```

```
    display: flex;
    align-items: center;
    justify-content: center;
    height: 100vh;
}

.container {
    background-color: white;
    padding: 25px;
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    width: 350px;
    text-align: center;
}

h2 {
    margin-bottom: 20px;
}

.form-group {
    position: relative;
    margin-bottom: 15px;
    text-align: left;
}

label {
    font-weight: bold;
}

input {
```

```
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    border: 1px solid #ccc;
    border-radius: 6px;
    font-size: 14px;
    outline: none;
    transition: border 0.3s ease-in-out;
}

input:focus {
    border-color: #007bff;
}

.password-container {
    position: relative;
}

.toggle-password {
    position: absolute;
    right: 10px;
    top: 50%;
    transform: translateY(-50%);
    cursor: pointer;
    font-size: 16px;
    color: #777;
}

.valid {
    border: 2px solid green !important;
```

```
}

.invalid {
    border: 2px solid red !important;
}

.error {
    color: red;
    font-size: 12px;
    margin-top: 3px;
    height: 14px;
}

button {
    background-color: #007bff;
    color: white;
    padding: 12px;
    border: none;
    border-radius: 6px;
    cursor: pointer;
    width: 100%;
    margin-top: 10px;
    font-size: 16px;
    opacity: 0.5;
    transition: background-color 0.3s ease-in-out, opacity 0.3s ease-in-out;
}

button:hover {
    background-color: #0056b3;
}
```

```
        button:enabled {
            opacity: 1;
        }
    </style>
</head>
<body>

    <div class="container">
        <h2>Register</h2>
        <form id="registrationForm">
            <div class="form-group">
                <label for="username">Username:</label>
                <input type="text" id="username" name="username" required>
                <p class="error" id="usernameError"></p>
            </div>
            <div class="form-group password-container">
                <label for="password">Password:</label>
                <input type="password" id="password" name="password" required>
                <span class="toggle-password" onclick="togglePassword('password', this)">👁</span>
                <p class="error" id="passwordError"></p>
            </div>
            <div class="form-group password-container">
                <label for="confirmPassword">Confirm Password:</label>
                <input type="password" id="confirmPassword" name="confirmPassword" required>
                <span class="toggle-password" onclick="togglePassword('confirmPassword', this)">👁</span>
                <p class="error" id="confirmPasswordError"></p>
            </div>
            <button type="submit" id="registerButton" disabled>Register</button>
        </form>
```

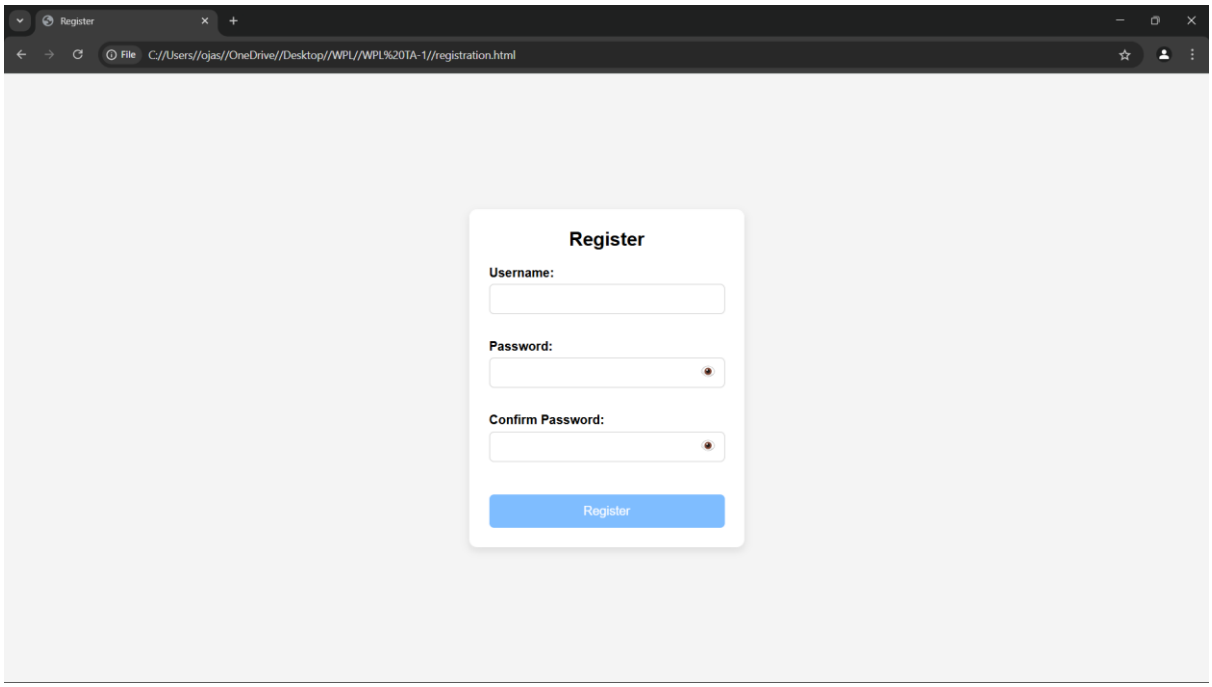
```
    <p id="successMessage" style="color: green; display: none;">Registration successful! Redirecting...</p>
</div>

<script>
    const form = document.getElementById("registrationForm");
    const usernameField = document.getElementById("username");
    const passwordField = document.getElementById("password");
    const confirmPasswordField = document.getElementById("confirmPassword");
    const registerButton = document.getElementById("registerButton");

    const validationRules = {
        username: {
            regex: /^[a-zA-Z0-9]{5,}$/,
            errorMsg: "Username must be at least 5 characters and contain only letters and numbers."
        },
        password: {
            regex: /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{6,16}$/,
            errorMsg: "Password must be 6-16 characters, with uppercase, lowercase, number, and special character."
        }
    };

    function validateField(field, rule) {
        const value = field.value.trim();
        const errorElement = document.getElementById(field.id + "Error");

        if (rule.regex.test(value)) {
            field.classList.add("valid");
            field.classList.remove("invalid");
            errorElement.textContent = "";
            return true;
```

```
    } else {
        field.classList.add("invalid");
        field.classList.remove("valid");
        errorElement.textContent = rule.errorMsg;
        return false;
    }
}

function validateConfirmPassword() {
    const password = passwordField.value;
    const confirmPassword = confirmPasswordField.value;
    const errorElement = document.getElementById("confirmPasswordError");

    if (confirmPassword === password && confirmPassword !== "") {
        confirmPasswordField.classList.add("valid");
        confirmPasswordField.classList.remove("invalid");
        errorElement.textContent = "";
        return true;
    } else {
        confirmPasswordField.classList.add("invalid");
        confirmPasswordField.classList.remove("valid");
        errorElement.textContent = "Passwords do not match.";
        return false;
    }
}

function validateForm() {
    const isUsernameValid = validateField(usernameField, validationRules.username);
    const isPasswordValid = validateField(passwordField, validationRules.password);
    const isConfirmPasswordValid = validateConfirmPassword();
```

```
        registerButton.disabled = !(isUsernameValid && isPasswordValid && isConfirmPasswordValid);
    }

    function togglePassword(fieldId, icon) {
        const field = document.getElementById(fieldId);
        if (field.type === "password") {
            field.type = "text";
            icon.textContent = "🙈";  // Hide Password Icon
        } else {
            field.type = "password";
            icon.textContent = "👁";  // Show Password Icon
        }
    }

    usernameField.addEventListener("input", validateForm);
    passwordField.addEventListener("input", validateForm);
    confirmPasswordField.addEventListener("input", validateForm);

    form.addEventListener("submit", function (event) {
        event.preventDefault();
        document.getElementById("successMessage").style.display = "block";
        setTimeout(() => window.location.href = "homepage.html", 2000);
    });
</script>

</body>
</html>
```

Output:

F. registration page output:



Code:

G. login page:

code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }
```

```
form {
    background-color: white;
    padding: 25px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
    width: 300px;
}

h2 {
    margin-bottom: 15px;
    text-align: center;
}

label {
    display: block;
    margin-top: 10px;
    font-weight: bold;
}

input.box {
    width: 100%;
    padding: 8px;
    margin-top: 5px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

#show-pass {
    margin-top: 10px;
    background: none;
```

```
            border: none;
            color: blue;
            cursor: pointer;
            text-decoration: underline;
        }

        #submit-btn {
            margin-top: 15px;
            width: 100%;
            padding: 10px;
            background-color: #333;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            opacity: 0.5;
        }

        #submit-btn:enabled {
            opacity: 1;
        }

        .msg {
            margin-top: 10px;
            font-weight: bold;
            text-align: center;
        }
    </style>
</head>
<body>
```

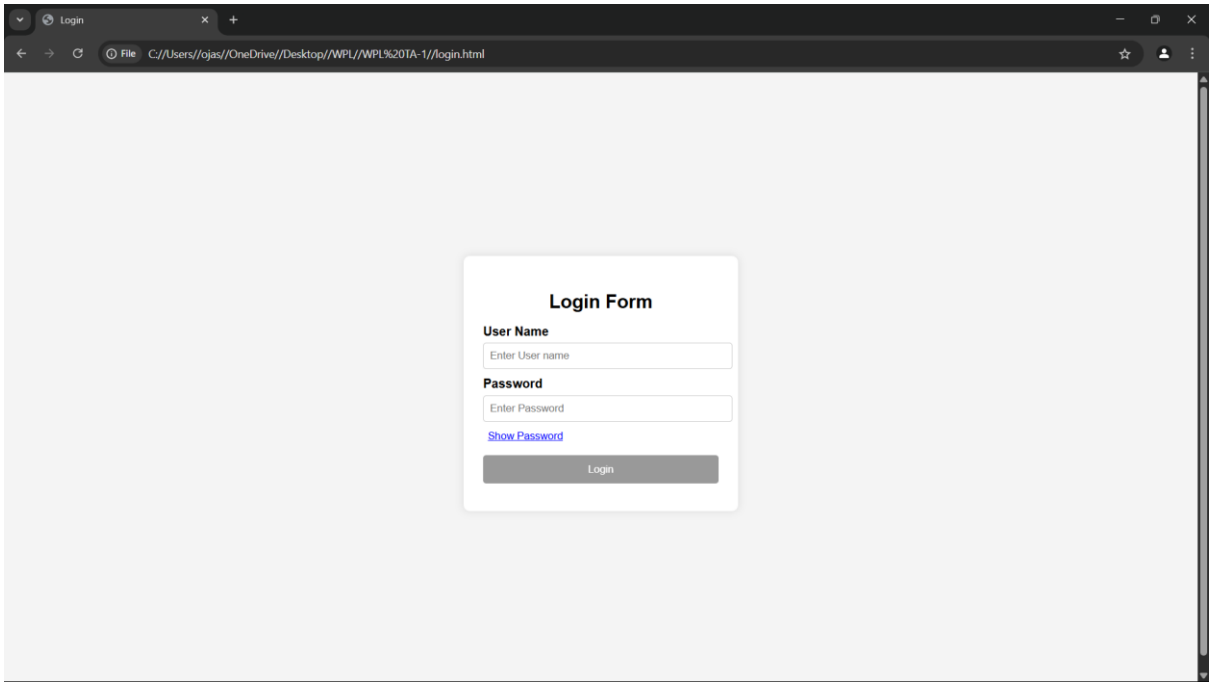
```
<form>
    <h2>Login Form</h2>
    <label for="username">User Name</label>
    <input type="text" class="box" placeholder="Enter User name" id="username" name="username">

    <label for="pass">Password</label>
    <input type="password" class="box" placeholder="Enter Password" id="pass" name="pass">

    <button id="show-pass">Show Password</button>

    <input type="submit" id="submit-btn" value="Login" disabled>

    <div class="msg"></div>
</form>

<script>
    const submit = document.getElementById('submit-btn');
    const msgElement = document.querySelector('.msg');
    const showPassBtn = document.getElementById('show-pass');
    const usernameInput = document.getElementById('username');
    const passwordInput = document.getElementById('pass');

    const validUser = "OjasUmate";
    const validPass = "Ojas@123";

    // Enable login button when both fields are filled
    usernameInput.addEventListener('input', validateForm);
    passwordInput.addEventListener('input', validateForm);

    function validateForm() {
```

```
    if (usernameInput.value.trim() && passwordInput.value.trim()) {
        submit.disabled = false;
    } else {
        submit.disabled = true;
    }
}

// Toggle password visibility
showPassBtn.addEventListener('click', function (e) {
    e.preventDefault();
    passwordInput.type = passwordInput.type === "password" ? "text" : "password";
    showPassBtn.textContent = passwordInput.type === "password" ? "Show Password" : "Hide Password";
});

// Handle form submission
submit.addEventListener('click', function (e) {
    e.preventDefault();

    let enteredUser = usernameInput.value.trim();
    let enteredPass = passwordInput.value;

    if (enteredUser === validUser && enteredPass === validPass) {
        msgElement.style.color = 'green';
        msgElement.textContent = 'Successfully logged in';

        localStorage.setItem('userDetails', JSON.stringify({ username: enteredUser }));

        setTimeout(() => {
            window.location.href = "homepage.html";
        }, 2000);
```

```
        } else {
            msgElement.style.color = 'red';
            msgElement.textContent = 'Invalid Username or Password';
        }
    });
  </script>
</body>
</html>
```

Output:

G. login page output:



Conclusion

The **Juice WRLD Albums Store** website design aims to offer fans of the late rapper a seamless and engaging platform where they can explore, purchase, and celebrate his musical legacy. Through careful planning and the use of intuitive design elements, we have created a layout that captures Juice WRLD's essence while providing a user-friendly experience.

By organizing the website into clear, well-defined sections—such as the Home, About, Albums, Testimonials, Contact, and Blog pages—the site ensures easy navigation and access to the essential content. The product pages highlight his music, albums, and related merchandise, with calls to action designed to encourage fans to engage with the content and make purchases.

Experiment No.3

Problem Statement:

Enhance the layout of the coffee shop website using CSS Grid for the home page.

Use CSS Grid to layout the menu/product items in a structured and style the menu categories with appropriate headings, spacing, separators, images, descriptions, and prices.

Theory:

Introduction to CSS Grid

CSS Grid Layout is a two-dimensional layout system optimized for web interfaces. Unlike Flexbox (which is one-dimensional), CSS Grid allows layout control both across rows and columns, making it ideal for complex responsive layouts, such as those found in e-commerce websites.
Using CSS Grid, designers and developers can create clean, consistent, and responsive page structures. This is particularly helpful for:

-

Landing pages with multiple content blocks (like a homepage).

-
-

Product listings in multiple categories (like an albums page).

-
-

Cart or gallery layouts with structured data display.

-

Why CSS Grid for this Website?

In a Juice WRLD album store, product presentation and layout are key to user satisfaction and engagement. Customers need to easily browse albums, compare products, and take quick actions. CSS Grid is used to:

-

Arrange albums in a neat grid (e.g., 3x3 or 4x4 grid).

-
-

Create sections like "Featured Albums", "Latest Releases", or "Exclusive Merch" in distinct, well-defined grid blocks.

-
-

Ensure consistent alignment of album images, titles, descriptions, and prices.

-
-

Support responsive design for mobile, tablet, and desktop screens.

-

1. Home Page Layout with CSS Grid

The homepage is structured into visually defined areas using CSS Grid:

-

A navigation header spanning full width.

-
-

A hero section with a large featured image or banner.

-
-

A three-column highlight section for featured albums or special offers.

-
-

A testimonial section laid out in a row.

-
-

A footer with contact info and social links.

Grid Benefits on Home Page:

Easy to define large areas and control layout positions.

Aligns different components (text, images, buttons) in a consistent way.

Makes the layout scalable and responsive without relying heavily on media queries.

2. Menu/Product Page Layout Using CSS Grid

This page displays the actual Juice WRLD albums and merchandise in a structured manner. Items are grouped into categories such as:

Albums (e.g., "Legends Never Die", "Goodbye & Good Riddance").

Vinyls.

Merchandise.

Special Editions.

Each product is displayed as a card, and all cards are arranged using CSS Grid for better responsiveness and visual balance.

Key Grid Features on Product Page:

-

Uniform item widths and spacing.

-
-

Grid gaps for breathing space between items.

-
-

Text (album name, description, price) aligned properly under images.

-
-

Easily allows 2, 3, or 4 columns depending on screen size.

-

Example CSS Grid Layout for Product Items:

```
css

CopyEdit

.products-grid {

  display: grid;

  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

  grid-gap: 30px;

  padding: 20px;

}
```

Each **product-card** inside this grid will have:

-

A product image.

-
-

A title.

-
-

A short description.

-
-

Price (highlighted).

-
-

"Add to Cart" button.

-

Additional Styling Concepts:

-

Category Headings: Styled with larger fonts, color backgrounds, or underlines to differentiate sections.

-
-

Separators: Thin horizontal lines or borders can visually divide different product categories.

-
-

Hover Effects: CSS transitions can enhance interactivity by highlighting cards or changing button styles on hover.

-
-

Responsive Design: CSS Grid's `auto-fit` and `minmax()` features allow the grid to adapt automatically to screen size, removing the need for complex media queries.

-

Mobile Responsiveness with CSS Grid

One of CSS Grid's biggest strengths is its responsive adaptability. The `grid-template-columns` property with `auto-fit` ensures that items stack or spread out based on available screen space.

Benefits for Mobile Users:

-

Grid automatically collapses to 1 or 2 columns.

-
-

Touch-friendly layout.

-
-

Ensures a smooth browsing experience.

-

Example CSS Grid Layout for Product Items:

```
.products-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
  grid-gap: 15px;
  padding: 10px;
}
.product-card {
  background: #fff;
  border: 1px solid #ddd;
  padding: 20px;
  text-align: center;
}
```

Additional Styling Concepts:

- **Category Headings**: Styled with larger fonts, color backgrounds, or underlines to differentiate sections.
- **Separators**: Thin horizontal lines or borders can visually divide different product categories.
- **Hover Effects**: CSS transitions can enhance interactivity by highlighting cards or changing button styles on hover.
- **Responsive Design**: CSS Grid's auto-fit and minmax() features allow the grid to adapt automatically to screen size, removing the need for complex media queries.

Mobile Responsiveness with CSS Grid

One of CSS Grid's biggest strengths is its **responsive adaptability**. The grid-template-columns property with auto-fit ensures that items stack or spread out based on available screen space.

Benefits for mobile users:

- Grid automatically collapses to 1 or 2 columns
- Touch-friendly layout
- Ensures a smooth browsing experience

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Juice WRLD Albums</title>
  <link rel="stylesheet" href="styles.css">
  <style>
    body {
      font-family: 'Arial', sans-serif;
      text-align: center;
      background: linear-gradient(to right, #1a1a1a, #333);
      color: white;
      margin: 0;
      padding: 0;
    }
    header {
      background: #111;
      color: white;
      padding: 20px 0;
      font-size: 24px;
      font-weight: bold;
```

```
    text-transform: uppercase;
    font-family: 'Courier New', Courier, monospace;
}
.nav-bar {
    display: flex;
    justify-content: center;
    gap: 20px;
    background: #222;
    padding: 10px;
}
.nav-bar a {
    color: #ff6600;
    text-decoration: none;
    font-size: 16px;
    transition: color 0.3s;
}
.nav-bar a:hover {
    color: #fff;
}
.album-container {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    gap: 20px;
    margin-top: 20px;
}
.album {
    background: #222;
    padding: 15px;
```

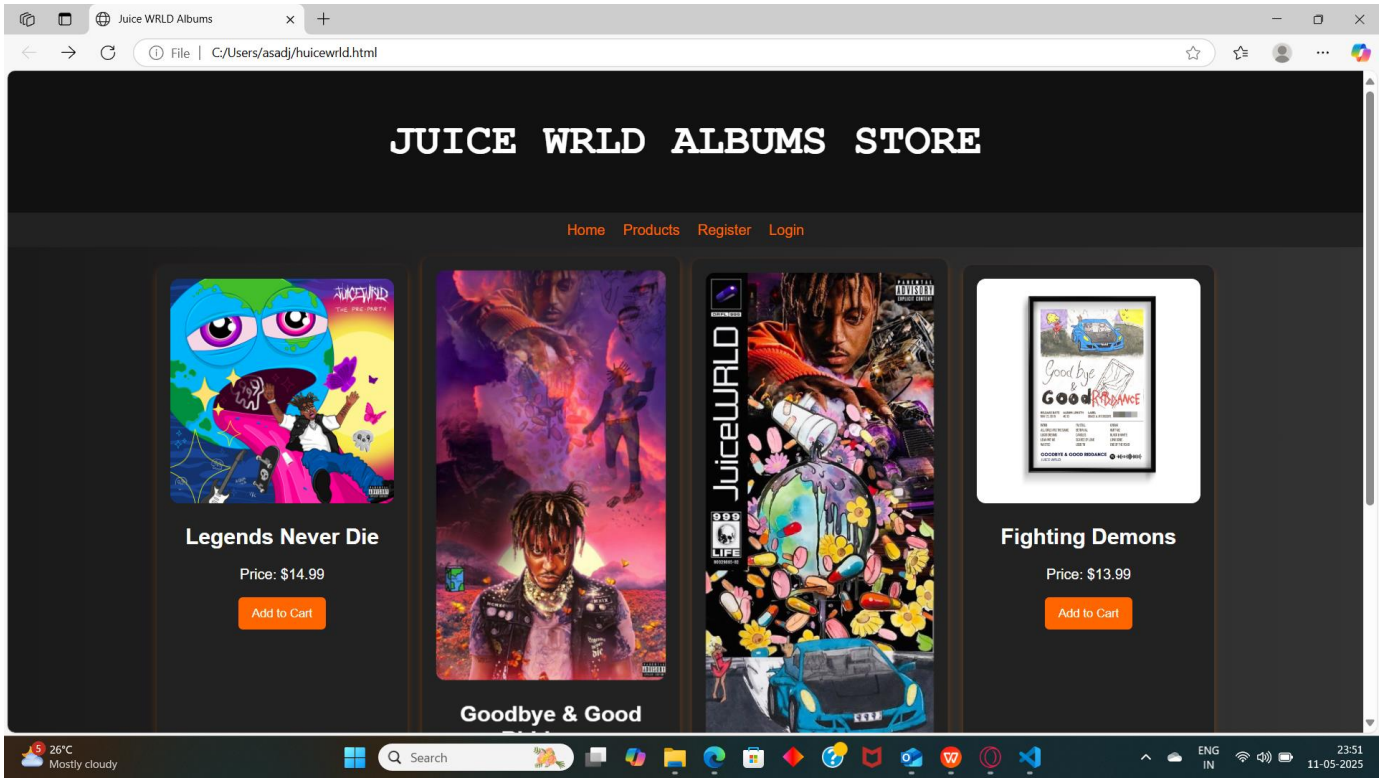
```
            border-radius: 10px;
            box-shadow: 0px 4px 8px rgba(255, 102, 0, 0.3);
            width: 250px;
            transition: transform 0.3s;
        }
        .album:hover {
            transform: scale(1.05);
        }
        .album img {
            width: 100%;
            border-radius: 10px;
        }
        .add-to-cart {
            background: #ff6600;
            color: white;
            padding: 10px 15px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
        .cart-container {
            margin-top: 30px;
            padding: 20px;
            background: #222;
            border-radius: 10px;
        }
    </style>
</head>
<body>
```

```
<header>
    <h1>Juice WRLD Albums Store</h1>
</header>
<nav class="nav-bar">
    <a href="index.html">Home</a>
    <a href="products.html">Products</a>
    <a href="register.html">Register</a>
    <a href="login.html">Login</a>
</nav>
<div class="album-container">
    <div class="album">
        <img src="https://i.scdn.co/image/ab67616d0000b273ed1cc06f375a38c942c0c3e5">
        <h2>Legends Never Die</h2>
        <p>Price: $14.99</p>
        <button class="add-to-cart" onclick="addToCart('Legends Never Die', 14.99)">Add to Cart</button>
    </div>
    <div class="album">
        <img src="https://i.redd.it/92p7e1pehrx51.jpg">
        <h2>Goodbye & Good Riddance</h2>
        <p>Price: $12.99</p>
        <button class="add-to-cart" onclick="addToCart('Goodbye & Good Riddance', 12.99)">Add to Cart</button>
    </div>
    <div class="album">
        <img src="https://i.pinimg.com/474x/46/3f/0e/463f0e4ca6804044d5bca227a467b455.jpg">
        <h2>Death Race for Love</h2>
        <p>Price: $15.99</p>
        <button class="add-to-cart" onclick="addToCart('Death Race for Love', 15.99)">Add to Cart</button>
    </div>
    <div class="album">
```

```
        <img src="https://themortalsoul.com/cdn/shop/files/44_e7cb768d-7188-4a1a-ba04-
81ed7ede243a.png?v=1735449715">
        <h2>Fighting Demons</h2>
        <p>Price: $13.99</p>
        <button class="add-to-cart" onclick="addToCart('Fighting Demons', 13.99)">Add to Cart</button>
    </div>
</div>

<div class="cart-container">
    <h2>Shopping Cart</h2>
    <ul id="cart-items"></ul>
    <h3>Total: $<span id="total-price">0.00</span></h3>
</div>

<script>
    let cart = [];
    let total = 0;

    function addToCart(albumName, price) {
        cart.push({ name: albumName, price: price });
        total += price;
        updateCart();
    }

    function updateCart() {
        const cartList = document.getElementById('cart-items');
        cartList.innerHTML = '';
        cart.forEach((item, index) => {
            const li = document.createElement('li');
```

```
            li.textContent = `${item.name} - $${item.price.toFixed(2)}`;

            const removeBtn = document.createElement('button');
            removeBtn.textContent = 'Remove';
            removeBtn.onclick = () => removeFromCart(index);

            li.appendChild(removeBtn);
            cartList.appendChild(li);
        });
        document.getElementById('total-price').innerText = total.toFixed(2);
    }

    function removeFromCart(index) {
        total -= cart[index].price;
        cart.splice(index, 1);
        updateCart();
    }
  </script>
</body>
</html>
```

Output



Conclusion

CSS Grid is an essential tool for crafting modern, responsive, and well-structured websites. In the case of a Juice WRLD album store, CSS Grid simplifies complex layout designs, enhances visual clarity, and creates a clean, user-friendly interface.

By using CSS Grid:

- **Home Page Layout**: The homepage is made visually appealing with clearly defined sections, allowing users to easily navigate through promotional banners, featured albums, and categories like "Latest Releases" and "Best Sellers."

- **Product Menu Organization**: The album menu is neatly structured, with product categories (like “Vinyl,” “Merchandise,” and “Special Editions”) displayed in a grid format for smooth exploration and easy access to details.

- **Responsive Design**: The website automatically adjusts for different screen sizes, providing a seamless browsing experience on desktops, tablets, and mobile devices without needing to write numerous media queries.

-

Consistent Layout and Spacing: CSS Grid ensures that the layout maintains uniform spacing between elements, providing a professional and polished look that complements Juice WRLD's branding.

-

In summary, CSS Grid is pivotal in enhancing the **visual appeal**, **usability**, and **responsiveness** of the Juice WRLD album store, ensuring the site is both functional and engaging for users. It helps maintain consistency in design while allowing the site to adapt across a variety of devices, making the browsing experience enjoyable for Juice WRLD fans everywhere.

Experiment No.4

When users visit an e-commerce website, their first impression is largely influenced by its visual appeal and usability. Clean, well-structured, and aesthetically pleasing interfaces can significantly boost user trust, ease of navigation, and engagement.

For a **Juice WRLD album store**, proper styling with CSS—such as adjusting margins, paddings, input designs, and using cohesive color schemes—plays a vital role in:

-

Making the content readable: Ensures that album titles, descriptions, and prices are easy to digest.

-
-

Providing structure: Helps organize content logically, whether it's the homepage, product listing, or checkout process.

-
-

Improving UX: Enhances user experience by making it intuitive and easy to browse through albums and merchandise.

-
-

Encouraging actions: Effective styling drives conversions, from purchasing an album to submitting contact forms or registering for exclusive updates.

-

Page-wise CSS Styling Theory

1. Cart Page

The cart page is crucial for users to review their purchases before proceeding to checkout. Proper styling ensures clarity and an action-driven design.

Key Styling Techniques:

-

Add padding around each album or item to create visual separation.

-
-

Use margins to space out product name, quantity, price, and action buttons like “Remove.”

-
-

Style input fields (quantity, update buttons) with soft borders, and ensure clickable areas are easy to interact with.

-
-

Highlight the **total amount** in bold font, using a distinct background for prominence.

-
-

Maintain consistent font sizes for price breakdowns and tax summaries.

-

Result: A well-structured cart page that minimizes confusion, ensures clarity, and maximizes the chances of completing the purchase.

---

2. About Us Page

The **About Us** page helps build a personal connection with your customers, providing insight into the brand’s mission and values.

Key Styling Techniques:

-

Use **line height** and **padding** for better readability.

-
-

Add white space between sections like “Our Story,” “Our Mission,” and “Meet the Team” to create a visually pleasant flow.

-
-

Use subtle background colors or horizontal lines to define each section.

-
-

Style images of the **team** with rounded borders and proper spacing to add a friendly vibe.

-
-

Highlight core values or quotes using boxes, grids, or bold typography for emphasis.

-

Result: A professional, inviting layout that builds trust and engages customers with the store’s story.

---

3. Contact Page

The **Contact Us** page should make it easy for users to reach out for support, queries, or feedback. A visually accessible and welcoming design is crucial.

Key Styling Techniques:

-

Style form **input fields** with equal width, padding, and rounded borders for a cohesive look.

-
-

Use **margin-bottom** between form fields to create visual separation.

-
-

Add **visual feedback** on focus (e.g., change in border color) to enhance interaction.

-
-

Style the **submit button** with hover effects or background color change to make it stand out.

-
-

Ensure the form is centrally aligned with balanced padding around all sides for better layout consistency.

-

Result: An aesthetically pleasing and user-friendly contact form that encourages engagement and communication.

---

4. Admin/User Registration Form

This form is essential for onboarding new users and administrators. A clean and secure-looking form enhances user trust and ensures a smooth registration process.

Key Styling Techniques:

-

Organize input fields in **logical groups** (e.g., personal info, password) to improve flow.

-
-

Include **labels** and **placeholders** for clarity.

-
-

Use consistent input sizes, padding, and spacing for all form fields.

-
-

Style the form card with **shadows**, **rounded borders**, and a **light background** to give it a polished and secure look.

-
-

Include **hover effects** for buttons and inline validation messages for immediate feedback.

-

Result: An intuitive, aesthetically appealing form that boosts confidence and encourages accurate and complete registration.

---

5. Admin/User Login Form

A streamlined login form is essential for users to access their accounts quickly and securely. Clarity and ease of use are key here.

Key Styling Techniques:

-

Center the login form on the page to draw focus.

-
-

Add sufficient **padding** inside the form container for balanced spacing.

-
-

Style **input fields** with proper spacing, highlighting them on focus for better visibility.

-
-

Use **subtle background colors** or semi-transparent overlays to make the form stand out.

-
-

Style **error messages** in **red** and **success messages** in **green** to provide immediate feedback.

-
-

Create a clear **visual hierarchy** by making the “Login” button larger and the “Forgot Password?” text smaller.

-

Result: A clean and efficient login page that builds user confidence and provides an easy-to-navigate entry point.

Code:

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Juice WRLD Albums</title>
    <link rel="stylesheet" href="styles.css">
    <style>
        body {
            font-family: 'Arial', sans-serif;
            text-align: center;
            background: linear-gradient(to right, #1a1a1a, #333);
            color: white;
            margin: 0;
            padding: 0;
        }
        header {
            background: #111;
            color: white;
            padding: 20px 0;
            font-size: 24px;
            font-weight: bold;
            text-transform: uppercase;
            font-family: 'Courier New', Courier, monospace;
        }
        .nav-bar {
            display: flex;
            justify-content: center;
            gap: 20px;
            background: #222;
            padding: 10px;
```

```
}
.nav-bar a {
    color: #ff6600;
    text-decoration: none;
    font-size: 16px;
    transition: color 0.3s;
}
.nav-bar a:hover {
    color: #fff;
}
.album-container {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    gap: 20px;
    margin-top: 20px;
}
.album {
    background: #222;
    padding: 15px;
    border-radius: 10px;
    box-shadow: 0px 4px 8px rgba(255, 102, 0, 0.3);
    width: 250px;
    transition: transform 0.3s;
}
.album:hover {
    transform: scale(1.05);
}
.album img {
    width: 100%;
```

```
            border-radius: 10px;
        }
        .add-to-cart {
            background: #ff6600;
            color: white;
            padding: 10px 15px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
        .cart-container {
            margin-top: 30px;
            padding: 20px;
            background: #222;
            border-radius: 10px;
        }
    </style>
</head>
<body>
    <header>
        <h1>Juice WRLD Albums Store</h1>
    </header>
    <nav class="nav-bar">
        <a href="index.html">Home</a>
        <a href="products.html">Products</a>
        <a href="register.html">Register</a>
        <a href="login.html">Login</a>
    </nav>
    <div class="album-container">
        <div class="album">
```

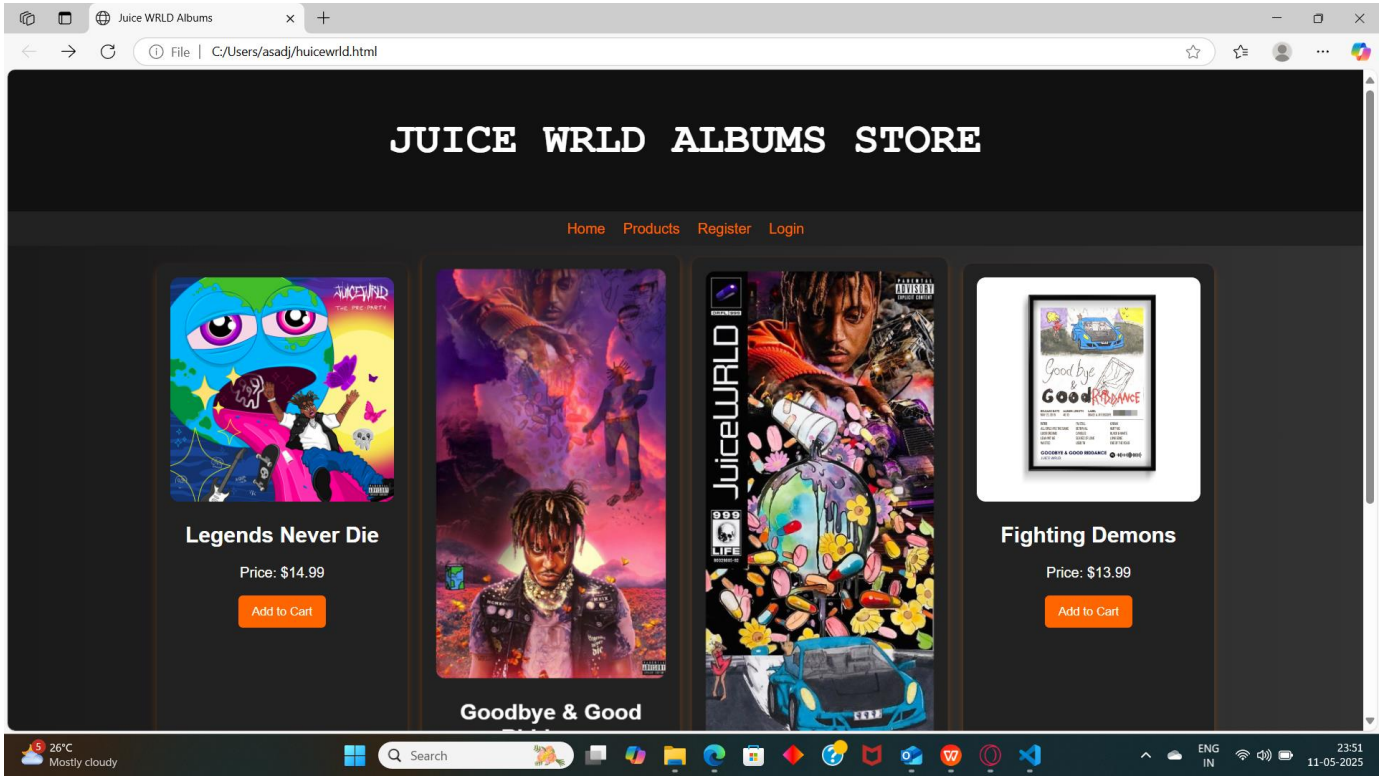
```
        <img src="https://i.scdn.co/image/ab67616d0000b273ed1cc06f375a38c942c0c3e5">
        <h2>Legends Never Die</h2>
        <p>Price: $14.99</p>
        <button class="add-to-cart" onclick="addToCart('Legends Never Die', 14.99)">Add to Cart</button>
    </div>
    <div class="album">
        <img src="https://i.redd.it/92p7e1pehrx51.jpg">
        <h2>Goodbye & Good Riddance</h2>
        <p>Price: $12.99</p>
        <button class="add-to-cart" onclick="addToCart('Goodbye & Good Riddance', 12.99)">Add to Cart</button>
    </div>
    <div class="album">
        <img src="https://i.pinimg.com/474x/46/3f/0e/463f0e4ca6804044d5bca227a467b455.jpg">
        <h2>Death Race for Love</h2>
        <p>Price: $15.99</p>
        <button class="add-to-cart" onclick="addToCart('Death Race for Love', 15.99)">Add to Cart</button>
    </div>
    <div class="album">
        <img src="https://themortalsoul.com/cdn/shop/files/44_e7cb768d-7188-4a1a-ba04-81ed7ede243a.png?v=1735449715">
        <h2>Fighting Demons</h2>
        <p>Price: $13.99</p>
        <button class="add-to-cart" onclick="addToCart('Fighting Demons', 13.99)">Add to Cart</button>
    </div>
</div>

<div class="cart-container">
    <h2>Shopping Cart</h2>
    <ul id="cart-items"></ul>
    <h3>Total: $<span id="total-price">0.00</span></h3>
</div>
```

```
<script>
    let cart = [];
    let total = 0;

    function addToCart(albumName, price) {
        cart.push({ name: albumName, price: price });
        total += price;
        updateCart();
    }

    function updateCart() {
        const cartList = document.getElementById('cart-items');
        cartList.innerHTML = '';
        cart.forEach((item, index) => {
            const li = document.createElement('li');
            li.textContent = `${item.name} - $${item.price.toFixed(2)}`;

            const removeBtn = document.createElement('button');
            removeBtn.textContent = 'Remove';
            removeBtn.onclick = () => removeFromCart(index);

            li.appendChild(removeBtn);
            cartList.appendChild(li);
        });
        document.getElementById('total-price').innerText = total.toFixed(2);
    }

    function removeFromCart(index) {
        total -= cart[index].price;
```

```
        cart.splice(index, 1);
        updateCart();
    }
</script>
</body>
</html>
```

Output:

page output:



Code:

registration page:

code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Register</title>
<style>
    * {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
        font-family: Arial, sans-serif;
    }

    body {
        background-color: #f4f4f4;
        display: flex;
        align-items: center;
        justify-content: center;
        height: 100vh;
    }

    .container {
        background-color: white;
        padding: 25px;
        border-radius: 10px;
        box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
        width: 350px;
        text-align: center;
    }

    h2 {
        margin-bottom: 20px;
    }
```

```
.form-group {
    position: relative;
    margin-bottom: 15px;
    text-align: left;
}

label {
    font-weight: bold;
}

input {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    border: 1px solid #ccc;
    border-radius: 6px;
    font-size: 14px;
    outline: none;
    transition: border 0.3s ease-in-out;
}

input:focus {
    border-color: #007bff;
}

.password-container {
    position: relative;
}

.toggle-password {
```

```
    position: absolute;
    right: 10px;
    top: 50%;
    transform: translateY(-50%);
    cursor: pointer;
    font-size: 16px;
    color: #777;
}

.valid {
    border: 2px solid green !important;
}

.invalid {
    border: 2px solid red !important;
}

.error {
    color: red;
    font-size: 12px;
    margin-top: 3px;
    height: 14px;
}

button {
    background-color: #007bff;
    color: white;
    padding: 12px;
    border: none;
    border-radius: 6px;
```

```
            cursor: pointer;
            width: 100%;
            margin-top: 10px;
            font-size: 16px;
            opacity: 0.5;
            transition: background-color 0.3s ease-in-out, opacity 0.3s ease-in-out;
        }

        button:hover {
            background-color: #0056b3;
        }

        button:enabled {
            opacity: 1;
        }
    </style>
</head>
<body>

    <div class="container">
        <h2>Register</h2>
        <form id="registrationForm">
            <div class="form-group">
                <label for="username">Username:</label>
                <input type="text" id="username" name="username" required>
                <p class="error" id="usernameError"></p>
            </div>
            <div class="form-group password-container">
                <label for="password">Password:</label>
                <input type="password" id="password" name="password" required>
```

```
            <span class="toggle-password" onclick="togglePassword('password', this)">👁</span>
            <p class="error" id="passwordError"></p>
        </div>
        <div class="form-group password-container">
            <label for="confirmPassword">Confirm Password:</label>
            <input type="password" id="confirmPassword" name="confirmPassword" required>
            <span class="toggle-password" onclick="togglePassword('confirmPassword', this)">👁</span>
            <p class="error" id="confirmPasswordError"></p>
        </div>
        <button type="submit" id="registerButton" disabled>Register</button>
    </form>
    <p id="successMessage" style="color: green; display: none;">Registration successful! Redirecting...</p>
</div>

<script>
    const form = document.getElementById("registrationForm");
    const usernameField = document.getElementById("username");
    const passwordField = document.getElementById("password");
    const confirmPasswordField = document.getElementById("confirmPassword");
    const registerButton = document.getElementById("registerButton");

    const validationRules = {
        username: {
            regex: /^[a-zA-Z0-9]{5,}$/,
            errorMsg: "Username must be at least 5 characters and contain only letters and numbers."
        },
        password: {
            regex: /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{6,16}$/,
            errorMsg: "Password must be 6-16 characters, with uppercase, lowercase, number, and special character."
        }
```

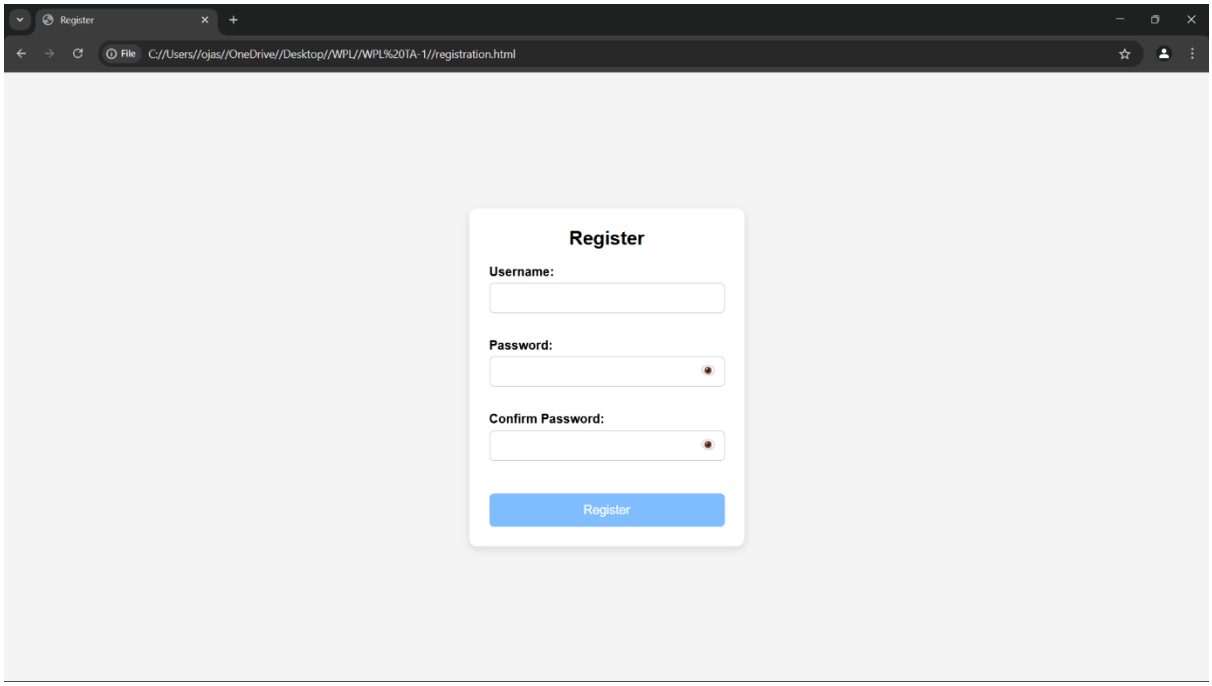
```
};

function validateField(field, rule) {
    const value = field.value.trim();
    const errorElement = document.getElementById(field.id + "Error");

    if (rule.regex.test(value)) {
        field.classList.add("valid");
        field.classList.remove("invalid");
        errorElement.textContent = "";
        return true;
    } else {
        field.classList.add("invalid");
        field.classList.remove("valid");
        errorElement.textContent = rule.errorMsg;
        return false;
    }
}

function validateConfirmPassword() {
    const password = passwordField.value;
    const confirmPassword = confirmPasswordField.value;
    const errorElement = document.getElementById("confirmPasswordError");

    if (confirmPassword === password && confirmPassword !== "") {
        confirmPasswordField.classList.add("valid");
        confirmPasswordField.classList.remove("invalid");
        errorElement.textContent = "";
        return true;
    } else {
```

```
        confirmPasswordField.classList.add("invalid");
        confirmPasswordField.classList.remove("valid");
        errorElement.textContent = "Passwords do not match.";
        return false;
    }
}

function validateForm() {
    const isUsernameValid = validateField(usernameField, validationRules.username);
    const isPasswordValid = validateField(passwordField, validationRules.password);
    const isConfirmPasswordValid = validateConfirmPassword();

    registerButton.disabled = !(isUsernameValid && isPasswordValid && isConfirmPasswordValid);
}

function togglePassword(fieldId, icon) {
    const field = document.getElementById(fieldId);
    if (field.type === "password") {
        field.type = "text";
        icon.textContent = "🙈";  // Hide Password Icon
    } else {
        field.type = "password";
        icon.textContent = "👁";  // Show Password Icon
    }
}

usernameField.addEventListener("input", validateForm);
passwordField.addEventListener("input", validateForm);
confirmPasswordField.addEventListener("input", validateForm);
```

```
        form.addEventListener("submit", function (event) {
            event.preventDefault();
            document.getElementById("successMessage").style.display = "block";
            setTimeout(() => window.location.href = "homepage.html", 2000);
        });
    </script>

</body>
</html>
```

Output:

registration page output:



Code:

login page:

code:

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Login</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }

        form {
            background-color: white;
            padding: 25px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
            width: 300px;
        }

        h2 {
            margin-bottom: 15px;
            text-align: center;
        }

        label {
            display: block;
            margin-top: 10px;
            font-weight: bold;
        }
```

```
input.box {
    width: 100%;
    padding: 8px;
    margin-top: 5px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

#show-pass {
    margin-top: 10px;
    background: none;
    border: none;
    color: blue;
    cursor: pointer;
    text-decoration: underline;
}

#submit-btn {
    margin-top: 15px;
    width: 100%;
    padding: 10px;
    background-color: #333;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    opacity: 0.5;
}
```

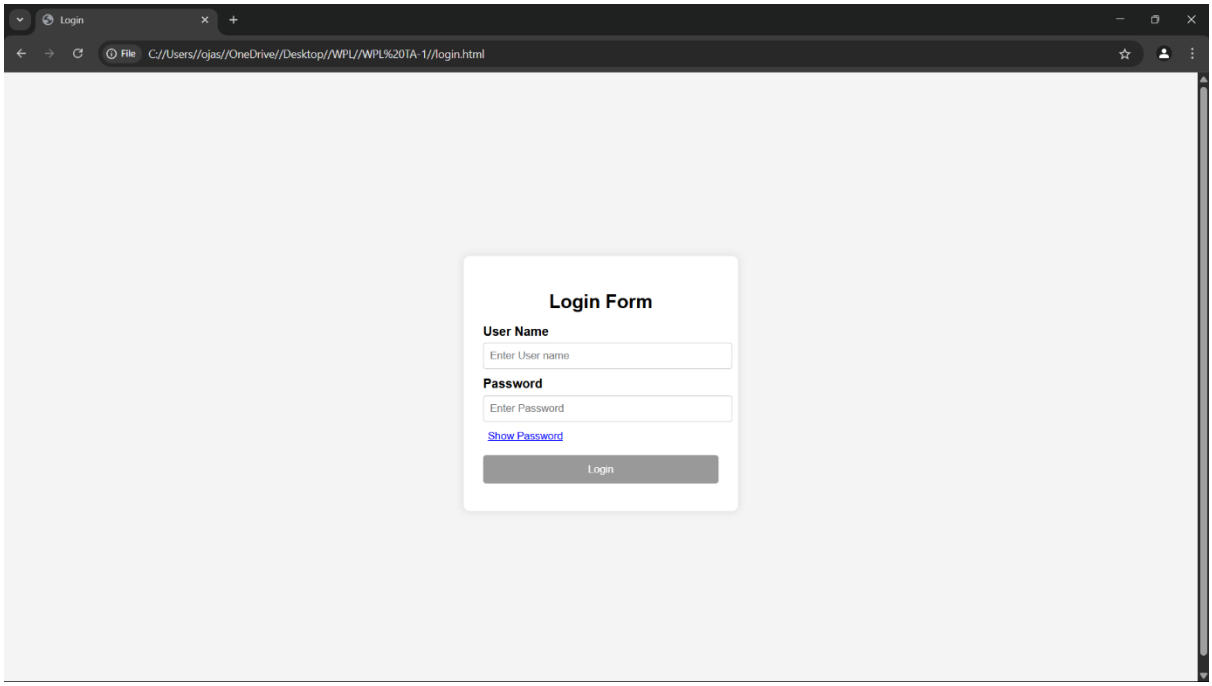
```
        #submit-btn:enabled {
            opacity: 1;
        }

        .msg {
            margin-top: 10px;
            font-weight: bold;
            text-align: center;
        }
    </style>
</head>
<body>
    <form>
        <h2>Login Form</h2>
        <label for="username">User Name</label>
        <input type="text" class="box" placeholder="Enter User name" id="username" name="username">

        <label for="pass">Password</label>
        <input type="password" class="box" placeholder="Enter Password" id="pass" name="pass">

        <button id="show-pass">Show Password</button>

        <input type="submit" id="submit-btn" value="Login" disabled>

        <div class="msg"></div>
    </form>

    <script>
        const submit = document.getElementById('submit-btn');
        const msgElement = document.querySelector('.msg');
```

```
const showPassBtn = document.getElementById('show-pass');
const usernameInput = document.getElementById('username');
const passwordInput = document.getElementById('pass');

const validUser = "OjasUmate";
const validPass = "Ojas@123";

// Enable login button when both fields are filled
usernameInput.addEventListener('input', validateForm);
passwordInput.addEventListener('input', validateForm);

function validateForm() {
    if (usernameInput.value.trim() && passwordInput.value.trim()) {
        submit.disabled = false;
    } else {
        submit.disabled = true;
    }
}

// Toggle password visibility
showPassBtn.addEventListener('click', function (e) {
    e.preventDefault();
    passwordInput.type = passwordInput.type === "password" ? "text" : "password";
    showPassBtn.textContent = passwordInput.type === "password" ? "Show Password" : "Hide Password";
});

// Handle form submission
submit.addEventListener('click', function (e) {
    e.preventDefault();
```

```
        let enteredUser = usernameInput.value.trim();
        let enteredPass = passwordInput.value;

        if (enteredUser === validUser && enteredPass === validPass) {
            msgElement.style.color = 'green';
            msgElement.textContent = 'Successfully logged in';

            localStorage.setItem('userDetails', JSON.stringify({ username: enteredUser }));

            setTimeout(() => {
                window.location.href = "homepage.html";
            }, 2000);
        } else {
            msgElement.style.color = 'red';
            msgElement.textContent = 'Invalid Username or Password';
        }
    });
  </script>
</body>
</html>
```

Output:

login page output:



Conclusion

CSS styling plays a pivotal role in ensuring that each page of the **Juice WRLD album store** looks professional and is easy to navigate. Proper styling enhances user experience, from browsing products to completing transactions. By using clear layouts, appealing design elements, and effective spacing, you can create a visually engaging, user-friendly platform that enhances both functionality and the overall shopping experience.

Experiment No.5

JavaScript Theory: User Registration, Login, Validation, and Cart Functionality

Introduction

In modern web development, **JavaScript** is a fundamental tool for creating interactive, dynamic, and user-friendly applications. For an e-commerce platform like the **Juice WRLD album store**, integrating features such as user registration, login, form validation, and cart functionality is crucial for providing a smooth and personalized shopping experience for users.

1. User Registration and Login Forms

User registration and login forms are essential for establishing user identity and ensuring personalized services on the Juice WRLD album store. JavaScript enhances the responsiveness and usability of these forms, ensuring the data is correctly validated before submission to the server.

Registration Form

The registration form allows new users to create an account by entering their personal details. Fields typically include full name, email address, password, confirm password, and optionally, phone number or address.

Key responsibilities of JavaScript in registration:

-

Ensuring all required fields are filled out

-
-

Verifying the email format using regular expressions

-
-

Checking that the password meets specific criteria (e.g., minimum length, use of special characters)

-
-

Ensuring the password and confirm password fields match

-
-

Providing real-time feedback if errors are found

-

Login Form

The login form allows returning users to access their accounts using their credentials.

Key responsibilities of JavaScript in login:

-

Ensuring the email and password fields are not empty

-
-

Validating the format of the email address

-
-

Matching input credentials against registered data (locally or via backend)

-
-

Redirecting the user to the dashboard or homepage upon successful authentication

-

2. JavaScript Form Validations

Form validation ensures data accuracy, completeness, and overall integrity. It enhances the user experience by providing immediate feedback and helps prevent incomplete or incorrect data submission.

Typical validation tasks include:

-

Ensuring mandatory fields are completed

-
-

Validating email address formats using regular expressions

-
-

Verifying password strength (length, use of special characters, etc.)

-
-

Ensuring passwords match

-
-

Displaying inline error messages for incorrect inputs

-

While client-side validation provides instant feedback, server-side validation should also be implemented for additional security.

3. Cart Functionality

The shopping cart is a crucial component for any e-commerce site. For the Juice WRLD album store, it allows users to review their selections, adjust quantities, and proceed to checkout.

Key JavaScript functionalities for the cart include:

-

Dynamically adding products to the cart

-
-

Updating item quantities and recalculating total prices

-
-

Removing items from the cart

-
-

Storing cart data in local storage or session storage for persistence

-
-

Dynamically rendering cart items using JavaScript’s DOM manipulation

-

By managing the cart structure as an array of objects in JavaScript, developers can efficiently track product details, prices, and overall totals, providing a seamless shopping experience.

Code:

F. registration page:

code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: Arial, sans-serif;
        }

        body {
            background-color: #f4f4f4;
            display: flex;
            align-items: center;
            justify-content: center;
            height: 100vh;
        }

        .container {
            background-color: white;
            padding: 25px;
            border-radius: 10px;
```

```
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    width: 350px;
    text-align: center;
}

h2 {
    margin-bottom: 20px;
}

.form-group {
    position: relative;
    margin-bottom: 15px;
    text-align: left;
}

label {
    font-weight: bold;
}

input {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    border: 1px solid #ccc;
    border-radius: 6px;
    font-size: 14px;
    outline: none;
    transition: border 0.3s ease-in-out;
}
```

```
input:focus {
    border-color: #007bff;
}

.password-container {
    position: relative;
}

.toggle-password {
    position: absolute;
    right: 10px;
    top: 50%;
    transform: translateY(-50%);
    cursor: pointer;
    font-size: 16px;
    color: #777;
}

.valid {
    border: 2px solid green !important;
}

.invalid {
    border: 2px solid red !important;
}

.error {
    color: red;
    font-size: 12px;
    margin-top: 3px;
```

```
            height: 14px;
        }

        button {
            background-color: #007bff;
            color: white;
            padding: 12px;
            border: none;
            border-radius: 6px;
            cursor: pointer;
            width: 100%;
            margin-top: 10px;
            font-size: 16px;
            opacity: 0.5;
            transition: background-color 0.3s ease-in-out, opacity 0.3s ease-in-out;
        }

        button:hover {
            background-color: #0056b3;
        }

        button:enabled {
            opacity: 1;
        }
    </style>
</head>
<body>

    <div class="container">
        <h2>Register</h2>
```

```
    <form id="registrationForm">
        <div class="form-group">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>
            <p class="error" id="usernameError"></p>
        </div>
        <div class="form-group password-container">
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required>
            <span class="toggle-password" onclick="togglePassword('password', this)">👁</span>
            <p class="error" id="passwordError"></p>
        </div>
        <div class="form-group password-container">
            <label for="confirmPassword">Confirm Password:</label>
            <input type="password" id="confirmPassword" name="confirmPassword" required>
            <span class="toggle-password" onclick="togglePassword('confirmPassword', this)">👁</span>
            <p class="error" id="confirmPasswordError"></p>
        </div>
        <button type="submit" id="registerButton" disabled>Register</button>
    </form>
    <p id="successMessage" style="color: green; display: none;">Registration successful! Redirecting...</p>
</div>

<script>
    const form = document.getElementById("registrationForm");
    const usernameField = document.getElementById("username");
    const passwordField = document.getElementById("password");
    const confirmPasswordField = document.getElementById("confirmPassword");
    const registerButton = document.getElementById("registerButton");
```

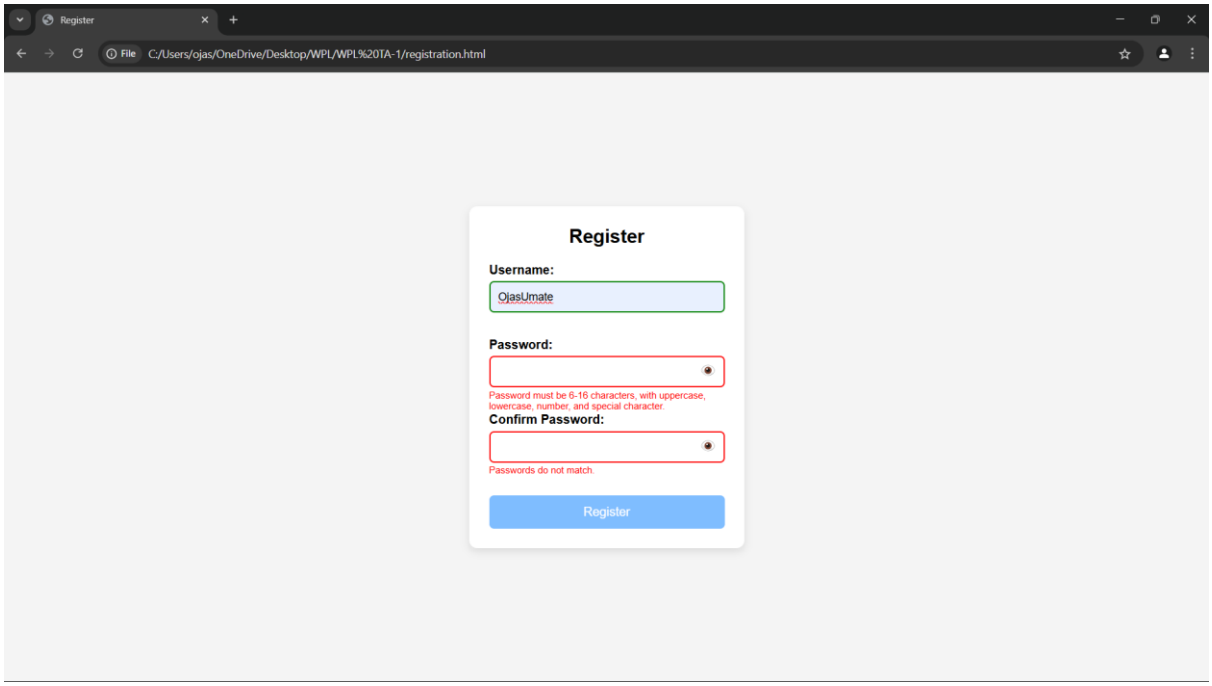
```
const validationRules = {
    username: {
        regex: /^[a-zA-Z0-9]{5,}$/,
        errorMsg: "Username must be at least 5 characters and contain only letters and numbers."
    },
    password: {
        regex: /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{6,16}$/,
        errorMsg: "Password must be 6-16 characters, with uppercase, lowercase, number, and special character."
    }
};

function validateField(field, rule) {
    const value = field.value.trim();
    const errorElement = document.getElementById(field.id + "Error");

    if (rule.regex.test(value)) {
        field.classList.add("valid");
        field.classList.remove("invalid");
        errorElement.textContent = "";
        return true;
    } else {
        field.classList.add("invalid");
        field.classList.remove("valid");
        errorElement.textContent = rule.errorMsg;
        return false;
    }
}

function validateConfirmPassword() {
    const password = passwordField.value;
```

```
    const confirmPassword = confirmPasswordField.value;
    const errorElement = document.getElementById("confirmPasswordError");

    if (confirmPassword === password && confirmPassword !== "") {
        confirmPasswordField.classList.add("valid");
        confirmPasswordField.classList.remove("invalid");
        errorElement.textContent = "";
        return true;
    } else {
        confirmPasswordField.classList.add("invalid");
        confirmPasswordField.classList.remove("valid");
        errorElement.textContent = "Passwords do not match.";
        return false;
    }
}

function validateForm() {
    const isUsernameValid = validateField(usernameField, validationRules.username);
    const isPasswordValid = validateField(passwordField, validationRules.password);
    const isConfirmPasswordValid = validateConfirmPassword();

    registerButton.disabled = !(isUsernameValid && isPasswordValid && isConfirmPasswordValid);
}

function togglePassword(fieldId, icon) {
    const field = document.getElementById(fieldId);
    if (field.type === "password") {
        field.type = "text";
        icon.textContent = "🙈";  // Hide Password Icon
    } else {
```

```
            field.type = "password";
            icon.textContent = "👁";  // Show Password Icon
        }
    }

    usernameField.addEventListener("input", validateForm);
    passwordField.addEventListener("input", validateForm);
    confirmPasswordField.addEventListener("input", validateForm);

    form.addEventListener("submit", function (event) {
        event.preventDefault();
        document.getElementById("successMessage").style.display = "block";
        setTimeout(() => window.location.href = "homepage.html", 2000);
    });
</script>

</body>
</html>
```

Output:

F. registration page output:



Code:

G. login page:

code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
        }

        form {
            background-color: white;
```

```
    padding: 25px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
    width: 300px;
}

h2 {
    margin-bottom: 15px;
    text-align: center;
}

label {
    display: block;
    margin-top: 10px;
    font-weight: bold;
}

input.box {
    width: 100%;
    padding: 8px;
    margin-top: 5px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

#show-pass {
    margin-top: 10px;
    background: none;
    border: none;
    color: blue;
```

```
            cursor: pointer;
            text-decoration: underline;
        }

        #submit-btn {
            margin-top: 15px;
            width: 100%;
            padding: 10px;
            background-color: #333;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            opacity: 0.5;
        }

        #submit-btn:enabled {
            opacity: 1;
        }

        .msg {
            margin-top: 10px;
            font-weight: bold;
            text-align: center;
        }
    </style>
</head>
<body>
    <form>
        <h2>Login Form</h2>
```

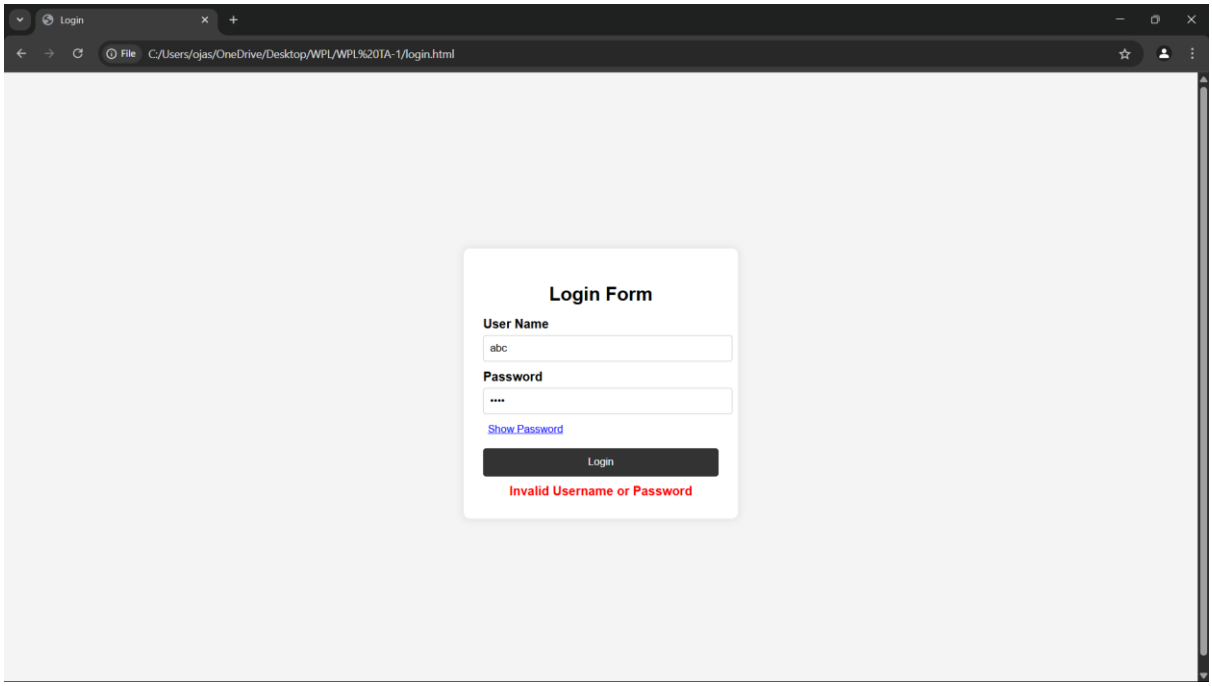
```
    <label for="username">User Name</label>
    <input type="text" class="box" placeholder="Enter User name" id="username" name="username">

    <label for="pass">Password</label>
    <input type="password" class="box" placeholder="Enter Password" id="pass" name="pass">

    <button id="show-pass">Show Password</button>

    <input type="submit" id="submit-btn" value="Login" disabled>

    <div class="msg"></div>
</form>

<script>
    const submit = document.getElementById('submit-btn');
    const msgElement = document.querySelector('.msg');
    const showPassBtn = document.getElementById('show-pass');
    const usernameInput = document.getElementById('username');
    const passwordInput = document.getElementById('pass');

    const validUser = "OjasUmate";
    const validPass = "Ojas@123";

    // Enable login button when both fields are filled
    usernameInput.addEventListener('input', validateForm);
    passwordInput.addEventListener('input', validateForm);

    function validateForm() {
        if (usernameInput.value.trim() && passwordInput.value.trim()) {
            submit.disabled = false;
```

```
    } else {
        submit.disabled = true;
    }
}

// Toggle password visibility
showPassBtn.addEventListener('click', function (e) {
    e.preventDefault();
    passwordInput.type = passwordInput.type === "password" ? "text" : "password";
    showPassBtn.textContent = passwordInput.type === "password" ? "Show Password" : "Hide Password";
});

// Handle form submission
submit.addEventListener('click', function (e) {
    e.preventDefault();

    let enteredUser = usernameInput.value.trim();
    let enteredPass = passwordInput.value;

    if (enteredUser === validUser && enteredPass === validPass) {
        msgElement.style.color = 'green';
        msgElement.textContent = 'Successfully logged in';

        localStorage.setItem('userDetails', JSON.stringify({ username: enteredUser }));

        setTimeout(() => {
            window.location.href = "homepage.html";
        }, 2000);
    } else {
        msgElement.style.color = 'red';
```

```
            msgElement.textContent = 'Invalid Username or Password';
        }
    });
  </script>
</body>
</html>
```

Output:

G. login page output:



Conclusion

JavaScript plays a crucial role in ensuring a smooth and engaging user experience for the **Juice WRLD album store**. By integrating robust functionalities such as user registration, login, form validation, and shopping cart management, the website can offer a seamless, personalized experience for its users.

Form validation ensures data integrity and provides immediate feedback, enhancing user confidence while interacting with the platform. The dynamic cart functionality allows for real-time updates, ensuring that users can easily manage their selections and proceed to checkout with ease.

Overall, JavaScript not only improves the interactivity and responsiveness of the site but also plays a critical part in making the online shopping experience enjoyable and efficient. Implementing these features ensures that the **Juice WRLD album store** stands out with a polished, user-centric interface that encourages engagement and conversion.

Experiment No.6

JavaScript Theory: Persistent Login and Cart Functionality using Web Storage API

Introduction

In modern web applications, providing a seamless user experience often requires maintaining user session states and data across different pages or after a page refresh. JavaScript's Web Storage API, which includes **localStorage** and **sessionStorage**, provides a lightweight and effective solution to store data on the client-side. For an e-commerce website like a **second-hand gaming console store**, utilizing these features can greatly improve usability by enabling persistent login sessions and preserving cart data, even after page reloads or temporary site exits.

1. Persistent Login using localStorage/sessionStorage

Login systems are essential for allowing users to securely access their accounts. JavaScript's Web Storage API offers a way to remember login statuses, reducing the need for users to log in repeatedly after each session.

- **localStorage**: Stores data indefinitely, persisting even after the browser is closed and reopened.
- **sessionStorage**: Stores data for the duration of the page session, meaning the data is cleared once the browser tab or window is closed.

Implementation Features:

- Upon successful login, JavaScript stores:
  - `userEmail`: the email or user ID of the logged-in user.
  - `isLoggedIn`: a boolean flag indicating the login status.
- On subsequent visits or page reloads:
  - JavaScript checks for these stored flags and either redirects the user to their dashboard or shows the login page if not logged in.
- **Logout functionality** clears the stored values, effectively ending the session.

Benefits:

-

Users don’t have to log in every time they visit the site, improving convenience and user experience.

-
-

Enhances session continuity, allowing users to return to their last activity without interruption.

-
-

Reduces the need for constant server-side validation, minimizing server load for smaller-scale or prototype applications.

-

2. Cart Data Management using localStorage

A shopping cart is one of the most important features in e-commerce websites. Users expect their cart contents to remain intact even after leaving the site or refreshing the page. **localStorage** allows the persistence of cart data, providing a more seamless and intuitive shopping experience.

Implementation Features:

-

Each time a user adds, removes, or updates a product in the cart:

-
    -

JavaScript serializes the cart array/object into JSON format and saves it to **localStorage**.

    -
-

On page load:

-
    -

JavaScript checks if cart data exists in **localStorage**.

    -
    -

If data is found, it is parsed and loaded into the cart view.

-

The cart remains persistent until the user explicitly clears it.

-

Benefits:

-

Prevents loss of cart data upon page reloads or accidental tab closure.

-
-

Enhances the shopping experience by maintaining cart state across sessions.

-
-

Provides a smoother transition for users, especially those who have not logged in or do not wish to create an account.

-

Use Cases Beyond the Syllabus (Advanced Learning):

These concepts often go beyond typical coursework and can help developers deepen their understanding:

-

Managing state with client-side storage: This is essential for creating dynamic, interactive applications that don't rely on server communication for every action.

-
-

Working with JSON and JavaScript objects: Serializing data and manipulating objects is critical in modern web development.

-
-

Handling user sessions in single-page or multi-page applications: Web Storage API allows for client-side session management without the need for a server-side backend, making it ideal for prototyping.

-

-

Creating realistic e-commerce simulations or portfolio projects: Implementing persistent login and cart functionality makes your projects more practical and engaging.

-

Conclusion

The Web Storage API is an invaluable tool in creating modern, user-friendly web applications. For a , utilizing **localStorage** and **sessionStorage** for login and cart data persistence significantly improves the user experience by making interactions more seamless and convenient. This functionality not only enhances usability but also encourages longer visits and smoother transitions between sessions.

Code:

code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Juice WRLD Albums</title>
    <link rel="stylesheet" href="styles.css">
    <style>
        body {
            font-family: 'Arial', sans-serif;
            text-align: center;
            background: linear-gradient(to right, #1a1a1a, #333);
            color: white;
            margin: 0;
            padding: 0;
        }
        header {
            background: #111;
            color: white;
```

```
    padding: 20px 0;
    font-size: 24px;
    font-weight: bold;
    text-transform: uppercase;
    font-family: 'Courier New', Courier, monospace;
}
.nav-bar {
    display: flex;
    justify-content: center;
    gap: 20px;
    background: #222;
    padding: 10px;
}
.nav-bar a {
    color: #ff6600;
    text-decoration: none;
    font-size: 16px;
    transition: color 0.3s;
}
.nav-bar a:hover {
    color: #fff;
}
.album-container {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    gap: 20px;
    margin-top: 20px;
}
.album {
```

```
            background: #222;
            padding: 15px;
            border-radius: 10px;
            box-shadow: 0px 4px 8px rgba(255, 102, 0, 0.3);
            width: 250px;
            transition: transform 0.3s;
        }
        .album:hover {
            transform: scale(1.05);
        }
        .album img {
            width: 100%;
            border-radius: 10px;
        }
        .add-to-cart {
            background: #ff6600;
            color: white;
            padding: 10px 15px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
        .cart-container {
            margin-top: 30px;
            padding: 20px;
            background: #222;
            border-radius: 10px;
        }
    </style>
</head>
```

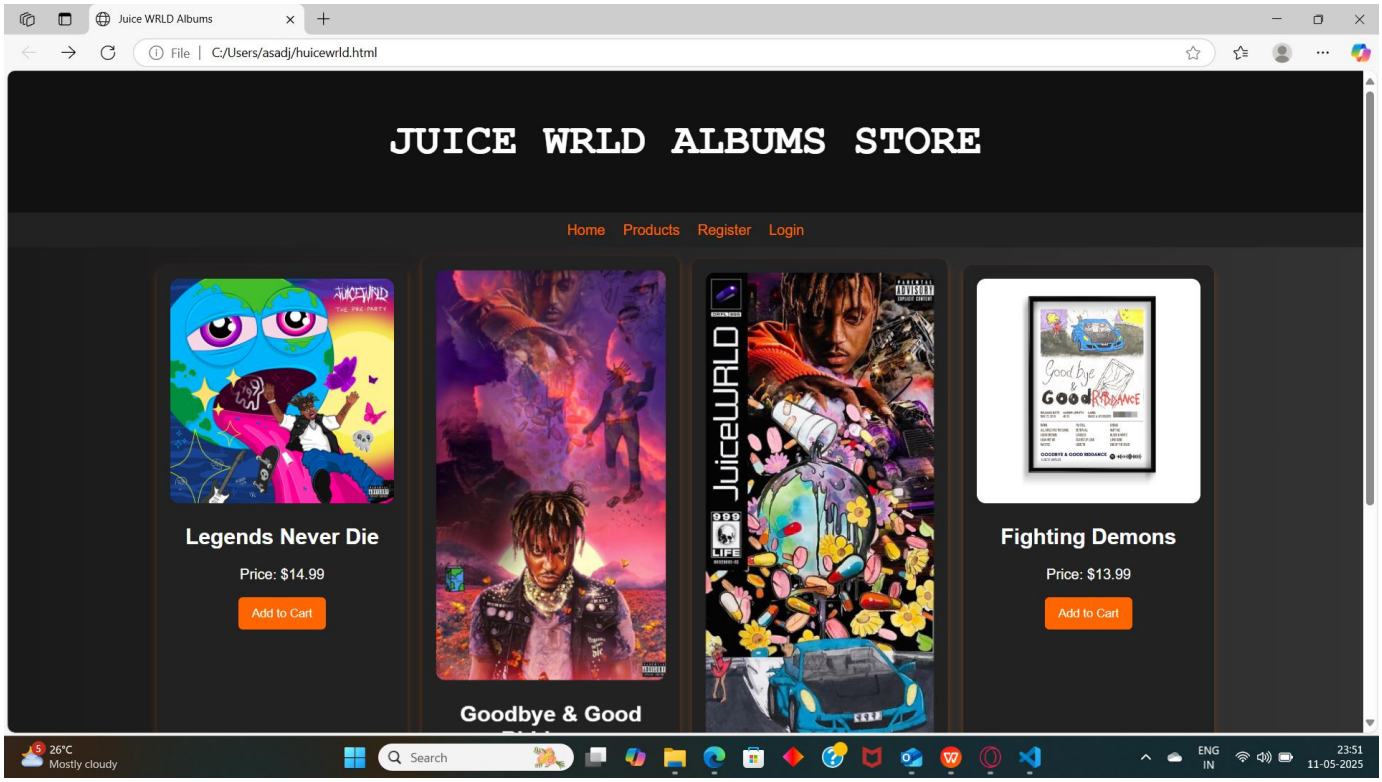
```
<body>
    <header>
        <h1>Juice WRLD Albums Store</h1>
    </header>
    <nav class="nav-bar">
        <a href="index.html">Home</a>
        <a href="products.html">Products</a>
        <a href="register.html">Register</a>
        <a href="login.html">Login</a>
    </nav>
    <div class="album-container">
        <div class="album">
            <img src="https://i.scdn.co/image/ab67616d0000b273ed1cc06f375a38c942c0c3e5">
            <h2>Legends Never Die</h2>
            <p>Price: $14.99</p>
            <button class="add-to-cart" onclick="addToCart('Legends Never Die', 14.99)">Add to Cart</button>
        </div>
        <div class="album">
            <img src="https://i.redd.it/92p7e1pehrx51.jpg">
            <h2>Goodbye & Good Riddance</h2>
            <p>Price: $12.99</p>
            <button class="add-to-cart" onclick="addToCart('Goodbye & Good Riddance', 12.99)">Add to Cart</button>
        </div>
        <div class="album">
            <img src="https://i.pinimg.com/474x/46/3f/0e/463f0e4ca6804044d5bca227a467b455.jpg">
            <h2>Death Race for Love</h2>
            <p>Price: $15.99</p>
            <button class="add-to-cart" onclick="addToCart('Death Race for Love', 15.99)">Add to Cart</button>
        </div>
        <div class="album">
```

```
                <img src="https://themortalsoul.com/cdn/shop/files/44_e7cb768d-7188-4a1a-ba04-81ed7ede243a.png?v=1735449715">
                <h2>Fighting Demons</h2>
                <p>Price: $13.99</p>
                <button class="add-to-cart" onclick="addToCart('Fighting Demons', 13.99)">Add to Cart</button>
            </div>
        </div>

        <div class="cart-container">
            <h2>Shopping Cart</h2>
            <ul id="cart-items"></ul>
            <h3>Total: $<span id="total-price">0.00</span></h3>
        </div>

        <script>
            let cart = [];
            let total = 0;

            function addToCart(albumName, price) {
                cart.push({ name: albumName, price: price });
                total += price;
                updateCart();
            }

            function updateCart() {
                const cartList = document.getElementById('cart-items');
                cartList.innerHTML = '';
                cart.forEach((item, index) => {
                    const li = document.createElement('li');
                    li.textContent = `${item.name} - $${item.price.toFixed(2)}`;
```

```
            const removeBtn = document.createElement('button');
            removeBtn.textContent = 'Remove';
            removeBtn.onclick = () => removeFromCart(index);

            li.appendChild(removeBtn);
            cartList.appendChild(li);
        });
        document.getElementById('total-price').innerText = total.toFixed(2);
    }

    function removeFromCart(index) {
        total -= cart[index].price;
        cart.splice(index, 1);
        updateCart();
    }
  </script>
</body>
</html>
```

Output:

A. Index/Home page output:



Conclusion

The Web Storage API is an invaluable tool in creating modern, user-friendly web applications. For a **second-hand gaming console website**, utilizing **localStorage** and **sessionStorage** for login and cart data persistence significantly improves the user experience by making interactions more seamless and convenient. This functionality not only enhances usability but also encourages longer visits and smoother transitions between sessions.

Experiment no.7

A. Develop a PHP script to handle user registration for the Coffee Shop website. The script should accept input from users for their name, email address, password, etc. (all required fields for registration).
B. Implement error handling to notify users of any issues during registration, such as validation errors.
C. Provide feedback to the user upon successful registration, either through a confirmation message or a redirect to a login page.

User registration is a fundamental component of web applications, particularly in e-commerce platforms like your second-hand gaming console website. PHP is widely used on the server side to handle form submissions, validate user inputs, interact with databases (like MySQL), and ensure secure data processing.

In this system, the registration form captures user details (name, email, password, etc.). Once submitted, the PHP script validates the inputs and then stores them securely into a database. To maintain security, user passwords are hashed before storage.

Core Elements of the PHP Registration Script:

1. Form Handling: Grabs data using $_POST.
2. Validation: Ensures fields are not empty and email is valid.
3. Password Hashing: Uses password_hash() to securely hash passwords.
4. Database Interaction: Uses MySQLi or PDO to store user data.
5. Error Handling: Displays messages for missing fields or registration failures.
6. User Feedback: Provides confirmation or redirection upon success.

CODE:-

```
<?php
// db_connect.php (include this file wherever needed)
$host = 'localhost';
$user = 'root';
$password = '';
$dbname = 'gaming_store';

$conn = new mysqli($host, $user, $password, $dbname);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

Registration:-

```
<?php
```

```
include 'db_connect.php';

$name = $email = $password = "";
$errors = [];

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Get input values and sanitize
    $name = trim($_POST["name"]);
    $email = trim($_POST["email"]);
    $password = trim($_POST["password"]);

    // Basic validation
    if (empty($name)) $errors[] = "Name is required.";
    if (empty($email) || !filter_var($email, FILTER_VALIDATE_EMAIL)) $errors[] = "Valid email is required.";
    if (empty($password) || strlen($password) < 6) $errors[] = "Password must be at least 6 characters.";

    // If no errors, proceed to store user
    if (empty($errors)) {
        $hashedPassword = password_hash($password, PASSWORD_BCRYPT);

        $stmt = $conn->prepare("INSERT INTO users (name, email, password) VALUES (?, ?, ?)");
        $stmt->bind_param("sss", $name, $email, $hashedPassword);

        if ($stmt->execute()) {
            echo "<p>Registration successful. <a href='login.html'>Click here to login</a>.</p>";
        } else {
            echo "<p>Error: " . $stmt->error . "</p>";
        }

        $stmt->close();
```

```
    } else {
        foreach ($errors as $error) {
            echo "<p style='color:red;'>$error</p>";
        }
    }

    $conn->close();
}
?>
```

Conclusion

Implementing user registration with PHP provides the backbone of user management in your website. By securely collecting, validating, and storing user data, you enable personalized experiences and functionalities such as login, saving favorites, or viewing past orders.

This system:

- Promotes user trust by securing sensitive data like passwords.
- Ensures data integrity through server-side validation.
- Enhances the user experience with real-time feedback and clear error handling.

Experiment 8

A. Develop a PHP script to handle user login for the Coffee Shop website. The script should accept input from users for their login credentials. (all required fields for login).
B. Provide feedback to the user upon successful login, either through a confirmation message or a redirect to a welcome page.
C. Implement error handling to notify users of login failures due to incorrect credentials or other errors.
D. Provide feedback to the user upon successful login, either through a welcome user name message or a redirect to a home page.

A **user login system** is a key feature of most online stores, especially for niche platforms like a **Juice WRLD Albums Store**. It allows fans and customers to securely access features like purchasing limited edition vinyls, saving favorite albums, or tracking past orders.

In PHP, login functionality typically includes:

Capturing login credentials via a form (email and password).

Validating the submitted input.

Matching credentials with stored data in a database.

-

-

Starting a session upon successful login.

-
-

Displaying a welcome message or redirecting to a dashboard.

-
-

Handling and displaying error messages for incorrect logins.

-

🔐 Security Aspects:

-

Password Hashing & Verification: During registration, passwords are securely stored using `password_hash()`. During login, PHP's `password_verify()` checks whether the entered password matches the stored hash.

-
-

Session Handling: PHP sessions are used to keep the user logged in across multiple pages of the site.

-

Code -

```
<?php

session_start();

include 'db_connect.php';


$email = $password = "";

$errors = [];


if ($_SERVER["REQUEST_METHOD"] == "POST") {

    $email = trim($_POST["email"]);
```

```
$password = trim($_POST["password"]);

// Basic validation
if (empty($email) || !filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $errors[] = "Please enter a valid email address.";
}
if (empty($password)) {
    $errors[] = "Please enter your password.";
}

if (empty($errors)) {
    $stmt = $conn->prepare("SELECT id, name, email, password FROM users WHERE email = ?");
    $stmt->bind_param("s", $email);
    $stmt->execute();

    $result = $stmt->get_result();

    if ($result && $result->num_rows === 1) {
        $user = $result->fetch_assoc();

        if (password_verify($password, $user['password'])) {
            $_SESSION["user_id"] = $user['id'];
            $_SESSION["user_name"] = $user['name'];
            $_SESSION["user_email"] = $user['email'];

            echo "<p>Welcome, <strong>" . htmlspecialchars($user['name']) . "</strong>! Redirecting to Juice WRLD dashboard...</p>";
            header("refresh:2;url=dashboard.php");
            exit();
        } else {
            $errors[] = "Incorrect password.";
```

```
            }
        } else {
            $errors[] = "No account found with that email.";
        }

        $stmt->close();
    }

    $conn->close();
}

// Display errors
foreach ($errors as $error) {
    echo "<p style='color:red;'>$error</p>";
}
?>
```

Experiment 9

A. Develop a PHP script that allows users to manage their shopping cart for an second hand gaming consoles website. The script should allow users to add items to their cart, view their cart contents, and remove items if

needed.
B. Develop a PHP script to manage the shopping cart for an second hand gaming consoles website using MySQL. This script should allow users to add items to their cart, view their cart contents, and remove items from the cart. The cart data should be stored in the MySQL database to allow persistence across sessions

Theory: PHP Shopping Cart System

A shopping cart is a core component of any e-commerce platform. It serves as a temporary storage space where users can collect and manage the items they wish to purchase. In the case of a second-hand gaming consoles website, where products can be unique and availability may be limited to single units, the shopping cart system plays an even more critical role.

Two Types of Cart Management Systems in PHP:

A. Session-Based Shopping Cart (Without MySQL)

This approach uses PHP sessions to temporarily store cart data in memory while the user is browsing. It is useful for fast prototyping and requires no database interaction.

Key Characteristics:

- Cart data is stored in $_SESSION.
- Data persists during the browsing session.
- No need to log in to use the cart.
- Items are lost if the session expires or the browser is closed.

Operations Supported:

- **Add to Cart**: Add items by storing product ID, name, quantity, and price in session.
- **View Cart**: Display the contents stored in session.
- **Remove from Cart**: Unset item by ID or index from the session.

Advantages:

- Simple to implement.
- No database overhead.

Limitations:

- Not persistent after session end.
- Not scalable for logged-in user experiences.

B. Database-Based Shopping Cart (With MySQL)

This is the professional and scalable approach where cart data is stored in a **MySQL database**. It allows cart contents to persist across user sessions, devices, and logins.

Key Characteristics:

- Each user has a unique cart identified by user ID.
- Cart contents are stored in a cart table, and optionally a cart_items table for item details.
- Requires user login or session management.

Operations Supported:

- **Add to Cart**: Insert or update records in the cart_items table.
- **View Cart**: Query database for all cart items belonging to a specific user.
- **Remove from Cart**: Delete an item from the database by item ID or cart ID.

Advantages:

- Cart is persistent and user-specific.
- Works across sessions and devices.
- Enables cart analytics and user behavior tracking.

Limitations:

- Requires more setup and error handling.
- Needs secure login system to link cart with user.

CODE:-

```
CREATE TABLE cart_items (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  product_id INT NOT NULL,
  product_name VARCHAR(255),
  quantity INT DEFAULT 1,
  price DECIMAL(10, 2),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Conclusion

A shopping cart system, whether session-based or database-driven, is essential for enhancing the user experience and improving sales on your juice wrld artist website .

When using PHP:

- **Session-based carts** offer fast and simple cart functionality, ideal for guest users.
- **MySQL-backed carts** provide reliable, persistent storage across sessions and devices—ideal for logged-in users and production-level systems.

For a fully functional and scalable website, the **MySQL-based cart** is highly recommended, as it:

- Improves user experience with persistent carts.
- Enables personalization and user analytics.
- Supports consistent item tracking (especially when each console unit is unique).

Experiment 10

A. Develop a PHP script to handle the checkout process for users who are ready to complete their purchase. The script should process the cart data and provide feedback to the user upon successful or failed checkout.
B. Develop a PHP script that processes the checkout process for users who are ready to complete their purchase, integrating the MySQL database for handling user and order information. The script should validate user input, process the cart data, and provide feedback upon successful or failed checkout.

Theory: PHP Checkout Process

The **checkout process** is the final and most crucial step in any e-commerce platform. It translates the user's cart into an official order, capturing necessary details such as billing, shipping, and payment, then recording it into the database for processing and fulfillment.

On a second-hand gaming console website, where products may be unique or limited, a **robust and accurate checkout system** ensures that stock integrity is maintained and customer satisfaction is upheld.

Two Approaches to Checkout

A. Session-Based Checkout (Without Database Order Management)

In this basic approach:

- All data is stored in the session ($_SESSION['cart']).
- On checkout, a confirmation message is shown.
- Useful for simple or demo applications. **Workflow:**

1. Retrieve cart from $_SESSION.
2. Validate input fields (name, email, address).
3. Show success or error message.
4. Clear cart after checkout.

Advantages:

- Quick to implement.
- Minimal setup required.

Limitations:

- Data not persistent.
- Not scalable or production-ready.
- No order history.

B. MySQL-Based Checkout System

This advanced and scalable approach:

- Stores order details in a MySQL database.
- Supports persistence, analytics, and back-end processing.
- Links orders to logged-in users.

Workflow:

1. Validate user session or login status.
2. Retrieve cart items from session or database.
3. Validate checkout fields (shipping info, contact).
4. Insert data into orders and order_items tables.
5. Display success/failure message.
6. Clear session cart.

Code:-

MYSQL Code

```
CREATE TABLE orders (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT,
  customer_name VARCHAR(255),
  customer_email VARCHAR(255),
  customer_address TEXT,
  total DECIMAL(10, 2),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE order_items (
  id INT AUTO_INCREMENT PRIMARY KEY,
  order_id INT,
  product_id INT,
  product_name VARCHAR(255),
  quantity INT,
  price DECIMAL(10, 2)
);
```

Checkout session:-

```
<?php
session_start();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (!isset($_SESSION['cart']) || empty($_SESSION['cart'])) {
        echo "Your cart is empty!";
        exit;
```

```
    }

    $name = $_POST['name'] ?? '';
    $email = $_POST['email'] ?? '';
    $address = $_POST['address'] ?? '';

    if (empty($name) || empty($email) || empty($address)) {
        echo "Please fill in all required fields.";
        exit;
    }

    echo "<h2>Order Summary</h2>";
    $total = 0;
    foreach ($_SESSION['cart'] as $item) {
        echo "{$item['name']} - Qty: {$item['quantity']} - ₹{$item['price']} <br>";
        $total += $item['quantity'] * $item['price'];
    }

    echo "<p>Total: ₹$total</p>";
    echo "<p>Thank you, $name! Your order has been placed.</p>";

    // Clear the cart
    unset($_SESSION['cart']);
} else {
    echo "Invalid request method.";
}
?>
```

MySQL-Based PHP Checkout Script:-

```
<?php
```

```
session_start();
$conn = new mysqli('localhost', 'root', '', 'gaming_store');

if ($conn->connect_error) {
    die("Database connection failed: " . $conn->connect_error);
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (!isset($_SESSION['cart']) || empty($_SESSION['cart'])) {
        echo "Your cart is empty.";
        exit;
    }

    $name = $_POST['name'] ?? '';
    $email = $_POST['email'] ?? '';
    $address = $_POST['address'] ?? '';
    $user_id = $_SESSION['user_id'] ?? 0;

    if (empty($name) || empty($email) || empty($address)) {
        echo "All fields are required.";
        exit;
    }

    $total = 0;
    foreach ($_SESSION['cart'] as $item) {
        $total += $item['quantity'] * $item['price'];
    }

    $stmt = $conn->prepare("INSERT INTO orders (user_id, customer_name, customer_email, customer_address, total) VALUES (?, ?, ?, ?, ?)");
    $stmt->bind_param("isssd", $user_id, $name, $email, $address, $total);
```

```
    if ($stmt->execute()) {
        $order_id = $stmt->insert_id;

        $itemStmt = $conn->prepare("INSERT INTO order_items (order_id, product_id, product_name, quantity, price) VALUES (?, ?, ?, ?, ?)");
        foreach ($_SESSION['cart'] as $item) {
            $itemStmt->bind_param("iisid", $order_id, $item['id'], $item['name'], $item['quantity'], $item['price']);
            $itemStmt->execute();
        }

        echo "<h2>Checkout Successful</h2>";
        echo "Thank you, <strong>$name</strong>. Your order ID is <strong>$order_id</strong>.<br>Total: ₹$total";
        unset($_SESSION['cart']);
    } else {
        echo "Checkout failed. Please try again.";
    }

    $stmt->close();
    $conn->close();
} else {
    echo "Invalid request.";
}
?>
```

```
mysql> SELECT * FROM login;
+----+---------------+-------------------------+-------------+-------+---------------------+
| id | username      | email                   | password    | role  | created_at          |
+----+---------------+-------------------------+-------------+-------+---------------------+
|  1 | ayesha_organic | ayesha@organicskin.com  | ayesha@123  | user  | 2025-05-12 12:19:10 |
|  2 | rahul_skincare | rahul@skinhealth.com    | rahul@456   | user  | 2025-05-12 12:19:10 |
|  3 | admin_organic  | admin@organicskin.com   | admin@789   | admin | 2025-05-12 12:19:10 |
|  4 | meera_glow     | meera@glownature.com    | meera@321   | user  | 2025-05-12 12:19:10 |
|  5 | natural_admin  | support@naturalcare.com | secure@admin | admin | 2025-05-12 12:19:10 |
+----+---------------+-------------------------+-------------+-------+---------------------+
5 rows in set (0.05 sec)
```

Conclusion

The checkout process is the most vital component of an e-commerce platform—it turns intent into action. For your juice wrld album website:

Use Case Importance:

- **Unique item inventory** means precise, real-time cart tracking is essential.
- **Persistence** through MySQL helps avoid loss of user choices and enables full order management.
- **Session-based approach** is useful in early development or guest checkout situations.

Session-Based Checkout Summary:

- Simple and fast.
- Best suited for demos or early-stage projects.
- Not ideal for multi-session or long-term tracking.

MySQL-Based Checkout Summary:

- Scalable and professional.
- Captures order history.
- Supports user-specific orders, data analytics, and future features like order cancellation or tracking.