**Front-End UI/UX Project**

**1. Title Page**

* **Project Title**: Music Playlist Organizer
* **Submitted By**: 
  + *Team Members-*

Blesson Baby (2460346)

Adi Sankar (2460308)

Chris Paul (2460350)

* + *College-E-mailid-*

[*blesson.baby@btech.christuniversity.in*](mailto:blesson.baby@btech.christuniversity.in)

[*adi.sankar@btech.christuniversity.in*](mailto:adi.sankar@btech.christuniversity.in)

[*chris.paul@btech.christuniversity.in*](mailto:chris.paul@btech.christuniversity.in)

* **Course**: *UI/UX Design Fundamentals*
* **Instructor Name**: *Dhiraj Alate.*
* **Institution**: *Christ University*
* **Date of Submission**: *26/09/2025*

**2. Abstract**

This project involved the development of a single-page, responsive, client-side Music Playlist Organizer Interface titled "Playlist Organizer". The application allows users to browse a library of curated playlists, apply complex filters based on genre, mood, artist, and duration, and manage tracks via a dedicated modal interface.

The design focuses on a clean, modern, dark-mode user interface, utilizing the Bootstrap 5 framework and custom CSS for responsiveness and a sleek aesthetic. The core functionality relies on jQuery for efficient DOM manipulation and dynamic content rendering (like renderGrid() and renderChips()) based on user-driven filters and search queries. The project successfully delivered an aesthetically modern and highly functional front-end application focused on efficient data display and user interaction for media organization.

**3. Objectives**

The primary objectives for this project were adapted to address the needs of a media organization application:

* To design and implement a user interface that allows simple and efficient browsing, searching, and filtering of a music playlist library.
* To create a dynamic and categorized filtering system (Offcanvas) based on attributes like Genre, Mood, and Artist, enabling multi-select filtering via interactive "filter chips".
* To utilize a responsive grid layout for displaying playlists (Card View) and a detailed, organized tracklist modal for individual playlist management, including basic audio controls (Play/Pause/Stop).
* To ensure the entire application is fully responsive across desktop, tablet, and mobile devices using the Bootstrap 5 framework.
* To implement a modern, dark-mode style using custom CSS variables (like --pri, --sec, and --card) to achieve a neo-brutalist "neo-neo" UI aesthetic.

**4. Scope of the Project**

The project focuses on developing a fully functional front-end, single-page web application using HTML, CSS, and JavaScript. The scope includes:

* Data Filtration: Implementing real-time text searching across playlist titles and tracks.
* Filtering Controls: Creation of genre, mood, artist, and duration filters in an offcanvas sidebar, allowing for combination filtering logic.
* Playlist View: Rendering a dynamic grid of interactive playlist cards that link to a detailed tracklist.
* Tracklist Modal: Displaying a dedicated modal with a detailed, sortable list of tracks for a selected playlist.
* User Interaction: Implementing audio play/pause controls within the tracklist modal.

**5. Tools & Technologies Used**

|  |  |
| --- | --- |
| Tool/Technology | Purpose |
| HTML5 | Core structure and semantic markup. |
| CSS3 | Custom styling and dark-mode theme. |
| Bootstrap 5.3 | Responsive layout, grid system, and components. |
| Chart.js 4.4.3 | Dynamic and data-driven chart generation. |
| jQuery 3.7.1 | Efficient DOM manipulation and form handling. |
| Bootstrap Icons | Clear and concise iconography. |
| Bootstrap 5.3 | Responsive layout, grid system, and components |

**6. HTML Structure Overview**

The application is structured around a single HTML file (

Untitled-4.html) using semantic tags and Bootstrap's container classes.

* Header/Navbar: A fixed top navigation bar featuring the project title ("Playlist Organizer") and primary search input and filter toggle button.
* Hero Section: A header section with a main descriptive title and quick filter chips.
* Main Container: A responsive container wraps the main grid content (#playlistGrid).
* Offcanvas Sidebar: A hidden sidebar (#filters) provides detailed filtering controls for Genre, Mood, Artist, and Duration.
* Modal (Tracklist): A large modal (#tracklistModal) is used to display the detailed list of tracks for any selected playlist.
* Footer: A simple footer providing copyright and author information

**7. CSS Styling Strategy**

The primary styling approach leverages a visually striking **dark-mode, media-player aesthetic** through custom CSS and utility classes.

* **CSS Variables (:root)**: Variables like --bg (background), --card (panel), --pri (primary color, purple/blue), and --sec (secondary color, cyan) manage the color palette, allowing for easy, centralized theme modification.
* **Neo-Brutalist Elements**: Custom classes like .playlist-card and .btn-neo utilize subtle gradients, borders, and box shadows to give elements a modern, tactile, slightly raised appearance.
* **Filter Chips**: Custom .filter-chip and .chip styles are implemented to create visually distinct, pill-shaped filtering elements, with an .active state clearly indicating the user's current selections.
* **Responsiveness**: Standard Bootstrap classes (row, col-lg-, etc.) are used for grid layout, and custom media queries ensure the Offcanvas filter is visible on large screens and collapses appropriately on smaller devices.

**8. Key Features**

* Responsive UI: Utilizes the Bootstrap grid for optimal viewing and interaction on all screen sizes.
* Multi-Filter System: Users can combine search terms with various category filters (Genre, Mood, Artist) to narrow down the displayed playlists.
* Dynamic Card Rendering: The main grid (#playlistGrid) is dynamically rendered (renderGrid()) based on the active filters and search query.
* Sorting Functionality: A dropdown allows users to sort playlists by criteria like "Featured," "Newest," or "Duration".
* Track Playback Simulation: The tracklist modal provides interactive buttons to initiate and stop audio playback (using the HTML <audio> element).
* Clear Filters: A dedicated button instantly resets all search queries, filters, and sorting parameters.

**9. Challenges Faced & Solutions**

|  |  |
| --- | --- |
| Challenge | Solution |
| Data Persistence | Used **Local Storage** to save and load all user data, ensuring data integrity across sessions. |
| Real-time Data Update | A single refreshUI() function updates all dashboard elements and charts after any data change. |
| Graphing Goal Progress | Integrated **Chart.js** with custom logic to calculate current totals and plot them against saved goals. |

**10. Outcome**

The Fitness Tracker Dashboard successfully demonstrates proficiency in building complex, data-driven front-end interfaces. By effectively combining Bootstrap, jQuery, and Chart.js, the project resulted in a functional, responsive, and visually appealing tool for personal fitness management. The use of client-side data persistence ensures the tool is immediately practical.

**11. Future Enhancements**

* User Authentication: Implement a full user login system to allow data to be stored on a server for multi-device access.
* Advanced Data Filtering: Add controls to filter the workout history and dashboard data by date range or workout type.
* Aesthetics: Integrate a light-mode/dark-mode toggle for user preference flexibility

**12. Code**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="utf-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>Music Playlist Organizer</title>

  <!-- Bootstrap 5 -->

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" />

  <!-- Icons -->

  <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.css" rel="stylesheet" />

  <!-- jQuery -->

  <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>

  <!-- Bootstrap JS -->

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

  <style>

    :root{

      --bg:#0e0f13;

      --card:#151822aa;

      --glass:#1b1f2b99;

      --pri:#7b5cff;

      --sec:#22d3ee;

      --txt:#e8eaf1;

      --muted:#9aa3b2;

      --chip:#24293a;

      --success:#22c55e;

      --danger:#ef4444;

    }

    body{

      background: radial-gradient(1200px 800px at 10% -20%, #1f1144 0%, transparent 60%),

                  radial-gradient(1000px 700px at 110% 10%, #04324a 0%, transparent 60%),

                  var(--bg);

      color: var(--txt);

      min-height: 100vh;

    }

    .navbar{

      background: linear-gradient(135deg, #0c0d12 0%, #131520 100%) !important;

      border-bottom: 1px solid #262a3a;

    }

    .brand-glow{

      background: linear-gradient(90deg, var(--pri), var(--sec));

      -webkit-background-clip: text;

      background-clip: text;

      color: transparent;

    }

    .search-bar{

      background: rgba(255,255,255,0.06);

      border: 1px solid rgba(255,255,255,0.08);

      border-radius: 16px;

      backdrop-filter: blur(10px);

      padding: 6px 12px;

      color: var(--txt);

    }

    .search-bar::placeholder{ color: var(--muted); }

    .playlist-card{

      background: linear-gradient(180deg, rgba(255,255,255,0.04), rgba(255,255,255,0.02));

      border: 1px solid rgba(255,255,255,0.08);

      border-radius: 20px;

      overflow: hidden;

      transition: transform .25s ease, border-color .25s ease, box-shadow .25s ease;

      box-shadow: 0 8px 28px rgba(0,0,0,0.25);

    }

    .playlist-card:hover{

      transform: translateY(-4px);

      border-color: rgba(123,92,255,0.4);

      box-shadow: 0 16px 40px rgba(123,92,255,0.12);

    }

    .cover{

      aspect-ratio: 1.6/1;

      object-fit: cover;

      width: 100%;

    }

    .chip{

      display: inline-flex;

      align-items: center;

      gap: 6px;

      font-size: 0.8rem;

      color: var(--muted);

      background: var(--chip);

      border: 1px solid rgba(255,255,255,0.08);

      padding: 6px 10px;

      border-radius: 999px;

    }

    .filter-chip{

      background: rgba(255,255,255,0.06);

      border: 1px solid rgba(255,255,255,0.08);

      color: var(--txt);

      padding: 8px 12px;

      border-radius: 999px;

      cursor: pointer;

      transition: all .2s ease;

    }

    .filter-chip.active{

      background: linear-gradient(90deg, rgba(123,92,255,0.26), rgba(34,211,238,0.26));

      border-color: rgba(123,92,255,0.45);

    }

    .offcanvas{

      background: rgba(21,24,34,0.9);

      backdrop-filter: blur(12px);

      border-left: 1px solid rgba(255,255,255,0.08);

      color: var(--txt);

    }

    .modal-content{

      background: rgba(16,19,28,0.92);

      border: 1px solid rgba(255,255,255,0.08);

      backdrop-filter: blur(12px);

      color: var(--txt);

      border-radius: 18px;

    }

    .track-row{

      display: grid;

      grid-template-columns: 32px 1fr 120px 70px 120px;

      gap: 12px;

      align-items: center;

      padding: 10px 12px;

      border-radius: 12px;

      transition: background .15s ease;

    }

    .track-row:hover{ background: rgba(255,255,255,0.05); }

    .btn-neo{

      background: linear-gradient(180deg, rgba(255,255,255,0.08), rgba(255,255,255,0.02));

      border: 1px solid rgba(255,255,255,0.12);

      color: var(--txt);

    }

    .btn-neo:hover{

      border-color: rgba(123,92,255,0.6);

      color: var(--txt);

    }

    .duration-badge{

      background: rgba(255,255,255,0.06);

      border: 1px solid rgba(255,255,255,0.08);

      padding: 4px 10px;

      border-radius: 999px;

      color: var(--muted);

      font-size: .85rem;

    }

    .stat{

      font-weight: 600;

      color: var(--txt);

    }

    .muted{ color: var(--muted); }

    .footer{

      color: var(--muted);

      border-top: 1px solid rgba(255,255,255,0.08);

    }

  </style>

</head>

<body>

  <!-- Navbar -->

  <nav class="navbar navbar-expand-lg sticky-top">

    <div class="container py-2">

      <a class="navbar-brand d-flex align-items-center gap-2" href="#">

        <i class="bi bi-music-note-beamed fs-4 text-info"></i>

        <span class="fw-bold brand-glow">Playlist Organizer</span>

      </a>

      <button class="navbar-toggler text-white" type="button" data-bs-toggle="collapse" data-bs-target="#nav">

        <span class="navbar-toggler-icon"></span>

      </button>

      <div id="nav" class="collapse navbar-collapse">

        <div class="ms-auto d-flex align-items-center gap-2">

          <div class="d-none d-md-flex align-items-center gap-2">

            <input id="searchInput" class="form-control form-control-lg search-bar" type="search" placeholder="Search playlists or tracks..." />

            <button id="openFilters" class="btn btn-neo" data-bs-toggle="offcanvas" data-bs-target="#filters"><i class="bi bi-sliders"></i> Filters</button>

          </div>

          <button class="btn btn-neo d-md-none" data-bs-toggle="offcanvas" data-bs-target="#filters"><i class="bi bi-sliders"></i></button>

        </div>

      </div>

    </div>

  </nav>

  <!-- Hero -->

  <header class="container py-4 py-md-5">

    <div class="row align-items-center">

      <div class="col-lg-7">

        <h1 class="display-5 fw-bold">Organize. Filter. Enjoy.</h1>

        <p class="lead muted">Browse curated playlists by genre, mood, artist, or duration, and manage tracks with a clean, responsive interface powered by Bootstrap 5 and jQuery for a seamless experience across devices.</p>

        <div class="d-flex gap-2 mt-2">

          <span class="chip"><i class="bi bi-stars text-warning"></i> Modern UI</span>

          <span class="chip"><i class="bi bi-phone text-success"></i> Responsive</span>

          <span class="chip"><i class="bi bi-lightning-charge text-info"></i> Fast Filters</span>

        </div>

      </div>

      <div class="col-lg-5 mt-4 mt-lg-0">

        <div class="p-3 p-md-4 rounded-4" style="background: var(--card); border:1px solid rgba(255,255,255,0.08);">

          <div class="d-flex align-items-center gap-3">

            <div class="flex-grow-1">

              <input id="searchInputSm" class="form-control form-control-lg search-bar" type="search" placeholder="Search playlists or tracks..." />

            </div>

            <button class="btn btn-neo" data-bs-toggle="offcanvas" data-bs-target="#filters"><i class="bi bi-sliders"></i></button>

          </div>

          <div class="d-flex gap-2 flex-wrap mt-3" id="quickGenres"></div>

        </div>

      </div>

    </div>

  </header>

  <!-- Filters Offcanvas -->

  <div class="offcanvas offcanvas-end" tabindex="-1" id="filters">

    <div class="offcanvas-header">

      <h5 class="offcanvas-title">Filters</h5>

      <button type="button" class="btn-close btn-close-white" data-bs-dismiss="offcanvas"></button>

    </div>

    <div class="offcanvas-body">

      <div class="mb-4">

        <h6 class="text-uppercase muted mb-2">Genre</h6>

        <div id="filterGenres" class="d-flex gap-2 flex-wrap"></div>

      </div>

      <div class="mb-4">

        <h6 class="text-uppercase muted mb-2">Mood</h6>

        <div id="filterMoods" class="d-flex gap-2 flex-wrap"></div>

      </div>

      <div class="mb-4">

        <h6 class="text-uppercase muted mb-2">Artist</h6>

        <div id="filterArtists" class="d-flex gap-2 flex-wrap"></div>

      </div>

      <div class="mb-4">

        <h6 class="text-uppercase muted mb-2">Duration</h6>

        <div id="filterDuration" class="d-flex gap-2 flex-wrap">

          <span class="filter-chip" data-duration="short">&lt; 30 min</span>

          <span class="filter-chip" data-duration="medium">30 - 60 min</span>

          <span class="filter-chip" data-duration="long">&gt; 60 min</span>

        </div>

      </div>

      <div class="d-flex gap-2">

        <button id="clearFilters" class="btn btn-outline-light">Clear</button>

        <button class="btn btn-primary" data-bs-dismiss="offcanvas">Apply</button>

      </div>

    </div>

  </div>

  <!-- Playlists Grid -->

  <main class="container pb-5">

    <div class="d-flex justify-content-between align-items-center mb-3">

      <div>

        <span class="muted">Showing</span> <span id="count" class="stat">0</span> <span class="muted">playlists</span>

      </div>

      <div>

        <select id="sortBy" class="form-select form-select-sm bg-transparent text-white" style="border-color: rgba(255,255,255,0.2);">

          <option value="featured">Featured</option>

          <option value="title">Title A-Z</option>

          <option value="duration">Duration</option>

          <option value="tracks">Tracks</option>

        </select>

      </div>

    </div>

    <div id="grid" class="row g-4"></div>

  </main>

  <!-- Playlist Details Modal -->

  <div class="modal fade" id="playlistModal" tabindex="-1">

    <div class="modal-dialog modal-dialog-centered modal-xl">

      <div class="modal-content">

        <div class="modal-header border-0">

          <div class="d-flex align-items-center gap-3">

            <img id="modalCover" class="rounded-3" style="width:90px;height:90px;object-fit:cover;" alt="cover" />

            <div>

              <h4 id="modalTitle" class="mb-1"></h4>

              <div class="muted" id="modalSubtitle"></div>

              <div class="d-flex gap-2 mt-2 flex-wrap">

                <span class="chip"><i class="bi bi-music-note"></i> <span id="modalTracks"></span> tracks</span>

                <span class="chip"><i class="bi bi-clock"></i> <span id="modalDuration"></span></span>

                <span class="chip"><i class="bi bi-tag"></i> <span id="modalGenres"></span></span>

                <span class="chip"><i class="bi bi-emoji-smile"></i> <span id="modalMoods"></span></span>

              </div>

            </div>

          </div>

          <button class="btn-close btn-close-white" data-bs-dismiss="modal"></button>

        </div>

        <div class="modal-body">

          <div class="table-responsive">

            <div id="trackList" class="d-grid gap-2"></div>

          </div>

        </div>

        <div class="modal-footer border-0">

          <div class="d-flex align-items-center gap-3 w-100">

            <div class="flex-grow-1">

              <audio id="audio" controls style="width:100%"></audio>

            </div>

            <button id="stopAudio" class="btn btn-outline-light"><i class="bi bi-stop"></i></button>

          </div>

        </div>

      </div>

    </div>

  </div>

  <!-- Footer -->

  <footer class="footer py-4">

    <div class="container d-flex justify-content-between flex-wrap gap-2">

      <div>Made with Bootstrap 5 and jQuery for a clean, responsive music organizer UI.</div>

      <div class="muted">Demo data for educational/internal use.</div>

    </div>

  </footer>

  <script>

    // Demo covers (data URLs placeholders)

    const demoCover = "https://images.unsplash.com/photo-1511379938547-c1f69419868d?q=80&w=1200&auto=format&fit=crop"; // public domain style

    const demoCover2 = "https://images.unsplash.com/photo-1470229722913-7c0e2dbbafd3?q=80&w=1200&auto=format&fit=crop";

    const demoCover3 = "https://images.unsplash.com/photo-1511671782779-c97d3d27a1d4?q=80&w=1200&auto=format&fit=crop";

    // Sample dataset (playlists + tracks) inspired by common music metadata structures

    // Structure influenced by community schemas (artist, track\_name, tags), adapted locally

    const data = [

      {

        id: "pl1",

        title: "Neon Nights",

        description: "Synthwave and chillwave for late-night focus.",

        cover: demoCover,

        genres: ["Synthwave","Electronic"],

        moods: ["Chill","Focus"],

        tracks: [

          { id:"t11", title:"Midnight Drive", artist:"Nova Arc", duration: 228, src:"https://cdn.pixabay.com/download/audio/2021/12/22/audio\_4a0e3f.mp3?filename=midnight-stroll-125701.mp3" },

          { id:"t12", title:"Glass Skyline", artist:"Nova Arc", duration: 196, src:"https://cdn.pixabay.com/download/audio/2022/03/15/audio\_04a6b0.mp3?filename=cyberpunk-moonlight-sonata-2000.mp3" },

          { id:"t13", title:"Vapor Bloom", artist:"Neon Park", duration: 242, src:"https://cdn.pixabay.com/download/audio/2021/09/25/audio\_6e7c5a.mp3?filename=deep-ambient-113985.mp3" }

        ]

      },

      {

        id: "pl2",

        title: "Sunrise Energy",

        description: "Upbeat pop and indie to kickstart the day.",

        cover: demoCover2,

        genres: ["Pop","Indie"],

        moods: ["Happy","Energetic"],

        tracks: [

          { id:"t21", title:"Bright Steps", artist:"Luma", duration: 184, src:"https://cdn.pixabay.com/download/audio/2021/08/08/audio\_1e5c60.mp3?filename=uplifting-pop-111303.mp3" },

          { id:"t22", title:"Paper Kites", artist:"Kalei", duration: 201, src:"https://cdn.pixabay.com/download/audio/2022/03/24/audio\_7d6ee7.mp3?filename=upbeat-pop-2009.mp3" },

          { id:"t23", title:"New Morning", artist:"Luma", duration: 215, src:"https://cdn.pixabay.com/download/audio/2023/02/28/audio\_4b8561.mp3?filename=sunny-day-140995.mp3" },

          { id:"t24", title:"Sky Lines", artist:"Harmona", duration: 175, src:"https://cdn.pixabay.com/download/audio/2022/03/01/audio\_0a3c3e.mp3?filename=good-time-ambient-21114.mp3" }

        ]

      },

      {

        id: "pl3",

        title: "Lo‑Fi Study",

        description: "Laid-back beats to focus and learn.",

        cover: demoCover3,

        genres: ["Lo‑Fi","Hip‑Hop"],

        moods: ["Calm","Focus"],

        tracks: [

          { id:"t31", title:"Coffee Rings", artist:"Rainy Alley", duration: 193, src:"https://cdn.pixabay.com/download/audio/2022/01/10/audio\_0a6190.mp3?filename=lofi-study-ambient-117997.mp3" },

          { id:"t32", title:"Notebook", artist:"Rainy Alley", duration: 205, src:"https://cdn.pixabay.com/download/audio/2021/11/15/audio\_d7e1d0.mp3?filename=lofi-chill-medium-ambient-114423.mp3" },

          { id:"t33", title:"Soft Focus", artist:"Blue Leaves", duration: 221, src:"https://cdn.pixabay.com/download/audio/2021/08/09/audio\_2f9c2e.mp3?filename=lofi-chill-111318.mp3" }

        ]

      }

    ];

    // Build quick lists

    const allGenres = [...new Set(data.flatMap(p => p.genres))];

    const allMoods = [...new Set(data.flatMap(p => p.moods))];

    const allArtists = [...new Set(data.flatMap(p => p.tracks.map(t=>t.artist)))];

    const state = {

      q: "",

      genres: new Set(),

      moods: new Set(),

      artists: new Set(),

      duration: null, // short <30, medium 30-60, long >60 (mins)

      sort: "featured"

    };

    const el = {

      grid: document.getElementById('grid'),

      count: document.getElementById('count'),

      sortBy: document.getElementById('sortBy'),

      searchInput: document.getElementById('searchInput'),

      searchInputSm: document.getElementById('searchInputSm'),

      audio: document.getElementById('audio'),

      stopAudio: document.getElementById('stopAudio'),

      modal: new bootstrap.Modal(document.getElementById('playlistModal')),

      modalCover: document.getElementById('modalCover'),

      modalTitle: document.getElementById('modalTitle'),

      modalSubtitle: document.getElementById('modalSubtitle'),

      modalTracks: document.getElementById('modalTracks'),

      modalDuration: document.getElementById('modalDuration'),

      modalGenres: document.getElementById('modalGenres'),

      modalMoods: document.getElementById('modalMoods'),

      trackList: document.getElementById('trackList'),

      quickGenres: document.getElementById('quickGenres'),

      filterGenres: document.getElementById('filterGenres'),

      filterMoods: document.getElementById('filterMoods'),

      filterArtists: document.getElementById('filterArtists'),

      filterDuration: document.getElementById('filterDuration'),

      clearFilters: document.getElementById('clearFilters'),

    };

    function secondsToHMS(s){

      const h = Math.floor(s/3600);

      const m = Math.floor((s%3600)/60);

      const ss = s%60;

      return (h>0? h+":" : "") + (m<10? "0"+m : m) + ":" + (ss<10? "0"+ss : ss);

    }

    function totalDuration(pl){

      return pl.tracks.reduce((sum,t)=>sum+t.duration,0);

    }

    function durationBucket(seconds){

      const mins = seconds/60;

      if (mins < 30) return "short";

      if (mins <= 60) return "medium";

      return "long";

    }

    function matchQuery(pl, q){

      if(!q) return true;

      const s = q.toLowerCase();

      if (pl.title.toLowerCase().includes(s)) return true;

      if (pl.description.toLowerCase().includes(s)) return true;

      if (pl.genres.some(g => g.toLowerCase().includes(s))) return true;

      if (pl.moods.some(m => m.toLowerCase().includes(s))) return true;

      if (pl.tracks.some(t => t.title.toLowerCase().includes(s) || t.artist.toLowerCase().includes(s))) return true;

      return false;

    }

    function applyFilters(){

      let list = data.slice();

      // query

      if (state.q) {

        list = list.filter(pl => matchQuery(pl, state.q));

      }

      // genres

      if (state.genres.size){

        list = list.filter(pl => [...state.genres].every(g => pl.genres.includes(g)));

      }

      // moods

      if (state.moods.size){

        list = list.filter(pl => [...state.moods].every(m => pl.moods.includes(m)));

      }

      // artists

      if (state.artists.size){

        list = list.filter(pl => {

          const artists = new Set(pl.tracks.map(t=>t.artist));

          return [...state.artists].every(a => artists.has(a));

        });

      }

      // duration

      if (state.duration){

        list = list.filter(pl => durationBucket(totalDuration(pl)) === state.duration);

      }

      // sort

      if (state.sort === "title"){

        list.sort((a,b)=> a.title.localeCompare(b.title));

      } else if (state.sort === "duration"){

        list.sort((a,b)=> totalDuration(b)-totalDuration(a));

      } else if (state.sort === "tracks"){

        list.sort((a,b)=> b.tracks.length - a.tracks.length);

      }

      return list;

    }

    function renderGrid(){

      const list = applyFilters();

      el.count.textContent = list.length.toString();

      el.grid.innerHTML = "";

      list.forEach(pl => {

        const mins = Math.round(totalDuration(pl)/60);

        const col = document.createElement('div');

        col.className = "col-12 col-sm-6 col-lg-4";

        col.innerHTML = `

          <div class="playlist-card h-100 d-flex flex-column">

            <img src="${pl.cover}" alt="${pl.title}" class="cover" />

            <div class="p-3 p-md-4 d-flex flex-column gap-2 flex-grow-1">

              <h5 class="mb-1">${pl.title}</h5>

              <div class="muted">${pl.description}</div>

              <div class="d-flex gap-2 flex-wrap mt-2">

                ${pl.genres.map(g=>`<span class="chip"><i class="bi bi-tag"></i> ${g}</span>`).join('')}

              </div>

              <div class="d-flex justify-content-between align-items-center mt-auto pt-2">

                <div class="d-flex align-items-center gap-2">

                  <span class="duration-badge"><i class="bi bi-clock"></i> ~${mins} min</span>

                  <span class="duration-badge"><i class="bi bi-music-note-list"></i> ${pl.tracks.length} tracks</span>

                </div>

                <button class="btn btn-primary btn-sm" data-open="${pl.id}">

                  <i class="bi bi-arrow-right-circle"></i> View

                </button>

              </div>

            </div>

          </div>

        `;

        el.grid.appendChild(col);

      });

      // bind open buttons

      el.grid.querySelectorAll('button[data-open]').forEach(btn=>{

        btn.addEventListener('click', ()=>{

          const id = btn.getAttribute('data-open');

          openPlaylist(id);

        });

      });

    }

    function renderChips(container, items, group){

      container.innerHTML = "";

      items.forEach(x=>{

        const span = document.createElement('span');

        span.className = "filter-chip";

        span.textContent = x;

        span.dataset[group] = x;

        if (state[group].has && state[group].has(x)) span.classList.add('active');

        span.addEventListener('click', ()=>{

          if (span.classList.contains('active')) {

            span.classList.remove('active');

            state[group].delete(x);

          } else {

            span.classList.add('active');

            state[group].add(x);

          }

          renderGrid();

        });

        container.appendChild(span);

      });

    }

    function renderQuickGenres(){

      el.quickGenres.innerHTML = "";

      allGenres.forEach(g=>{

        const b = document.createElement('button');

        b.className = "btn btn-neo btn-sm";

        b.innerHTML = `<i class="bi bi-tag"></i> ${g}`;

        b.addEventListener('click', ()=>{

          state.genres.clear();

          state.genres.add(g);

          renderChips(el.filterGenres, allGenres, "genres");

          renderGrid();

        });

        el.quickGenres.appendChild(b);

      });

    }

    function openPlaylist(id){

      const pl = data.find(p=>p.id===id);

      if (!pl) return;

      el.modalCover.src = pl.cover;

      el.modalTitle.textContent = pl.title;

      el.modalSubtitle.textContent = pl.description;

      el.modalTracks.textContent = pl.tracks.length.toString();

      el.modalDuration.textContent = secondsToHMS(totalDuration(pl));

      el.modalGenres.textContent = pl.genres.join(", ");

      el.modalMoods.textContent = pl.moods.join(", ");

      // track list

      el.trackList.innerHTML = "";

      pl.tracks.forEach((t, idx)=>{

        const row = document.createElement('div');

        row.className = "track-row";

        row.innerHTML = `

          <div class="text-center muted">${idx+1}</div>

          <div>

            <div class="fw-semibold">${t.title}</div>

            <div class="muted small">${t.artist}</div>

          </div>

          <div><span class="duration-badge">${secondsToHMS(t.duration)}</span></div>

          <div class="text-center">

            <button class="btn btn-sm btn-neo" data-play="${t.src}"><i class="bi bi-play-fill"></i></button>

          </div>

          <div class="text-end">

            <button class="btn btn-sm btn-outline-warning me-2" data-edit="${t.id}"><i class="bi bi-pencil"></i></button>

            <button class="btn btn-sm btn-outline-danger" data-remove="${t.id}"><i class="bi bi-trash"></i></button>

          </div>

        `;

        el.trackList.appendChild(row);

      });

      // bind play

      el.trackList.querySelectorAll('button[data-play]').forEach(b=>{

        b.addEventListener('click', ()=>{

          const src = b.getAttribute('data-play');

          el.audio.src = src;

          el.audio.play();

        });

      });

      // bind edit/remove demos

      el.trackList.querySelectorAll('button[data-edit]').forEach(b=>{

        b.addEventListener('click', ()=>{

          const tid = b.getAttribute('data-edit');

          alert("Edit track metadata (demo): " + tid);

        });

      });

      el.trackList.querySelectorAll('button[data-remove]').forEach(b=>{

        b.addEventListener('click', ()=>{

          const tid = b.getAttribute('data-remove');

          if (confirm("Remove track from playlist?")){

            const ix = pl.tracks.findIndex(x=>x.id===tid);

            if (ix>=0){ pl.tracks.splice(ix,1); openPlaylist(id); renderGrid(); }

          }

        });

      });

      el.modal.show();

    }

    function bindEvents(){

      // search

      const syncQuery = (val)=>{

        state.q = val.trim();

        el.searchInput.value = val;

        el.searchInputSm.value = val;

        renderGrid();

      };

      el.searchInput.addEventListener('input', e=> syncQuery(e.target.value));

      el.searchInputSm.addEventListener('input', e=> syncQuery(e.target.value));

      // sort

      el.sortBy.addEventListener('change', ()=>{

        state.sort = el.sortBy.value;

        renderGrid();

      });

      // duration chip group

      el.filterDuration.querySelectorAll('.filter-chip').forEach(chip=>{

        chip.addEventListener('click', ()=>{

          const dur = chip.getAttribute('data-duration');

          // toggle exclusive

          if (chip.classList.contains('active')){

            chip.classList.remove('active');

            state.duration = null;

          } else {

            el.filterDuration.querySelectorAll('.filter-chip').forEach(c=>c.classList.remove('active'));

            chip.classList.add('active');

            state.duration = dur;

          }

          renderGrid();

        });

      });

      // clear

      el.clearFilters.addEventListener('click', ()=>{

        state.q = "";

        state.genres.clear();

        state.moods.clear();

        state.artists.clear();

        state.duration = null;

        state.sort = "featured";

        el.searchInput.value = "";

        el.searchInputSm.value = "";

        el.sortBy.value = "featured";

        renderChips(el.filterGenres, allGenres, "genres");

        renderChips(el.filterMoods, allMoods, "moods");

        renderChips(el.filterArtists, allArtists, "artists");

        el.filterDuration.querySelectorAll('.filter-chip').forEach(c=>c.classList.remove('active'));

        renderGrid();

      });

      // stop audio

      el.stopAudio.addEventListener('click', ()=>{

        el.audio.pause();

        el.audio.currentTime = 0;

      });

    }

    // init

    function init(){

      renderChips(el.filterGenres, allGenres, "genres");

      renderChips(el.filterMoods, allMoods, "moods");

      renderChips(el.filterArtists, allArtists, "artists");

      renderQuickGenres();

      bindEvents();

      renderGrid();

    }

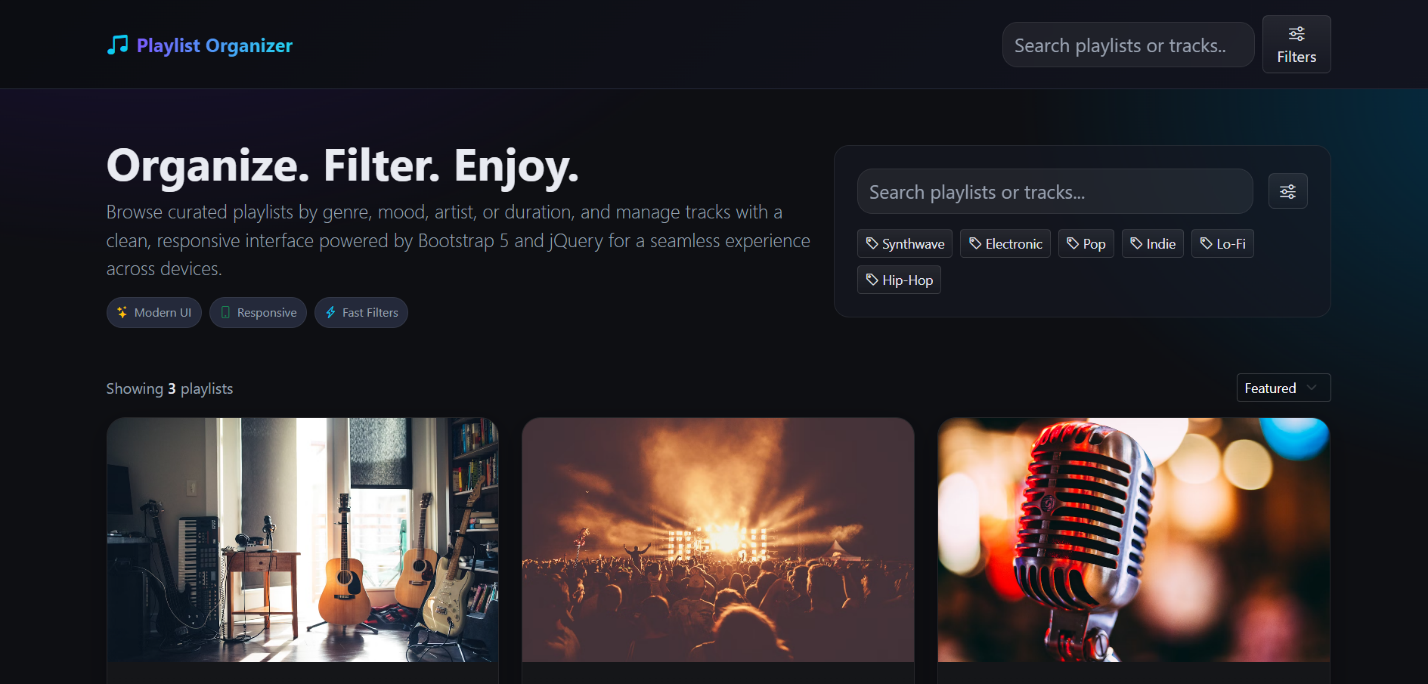
    init();

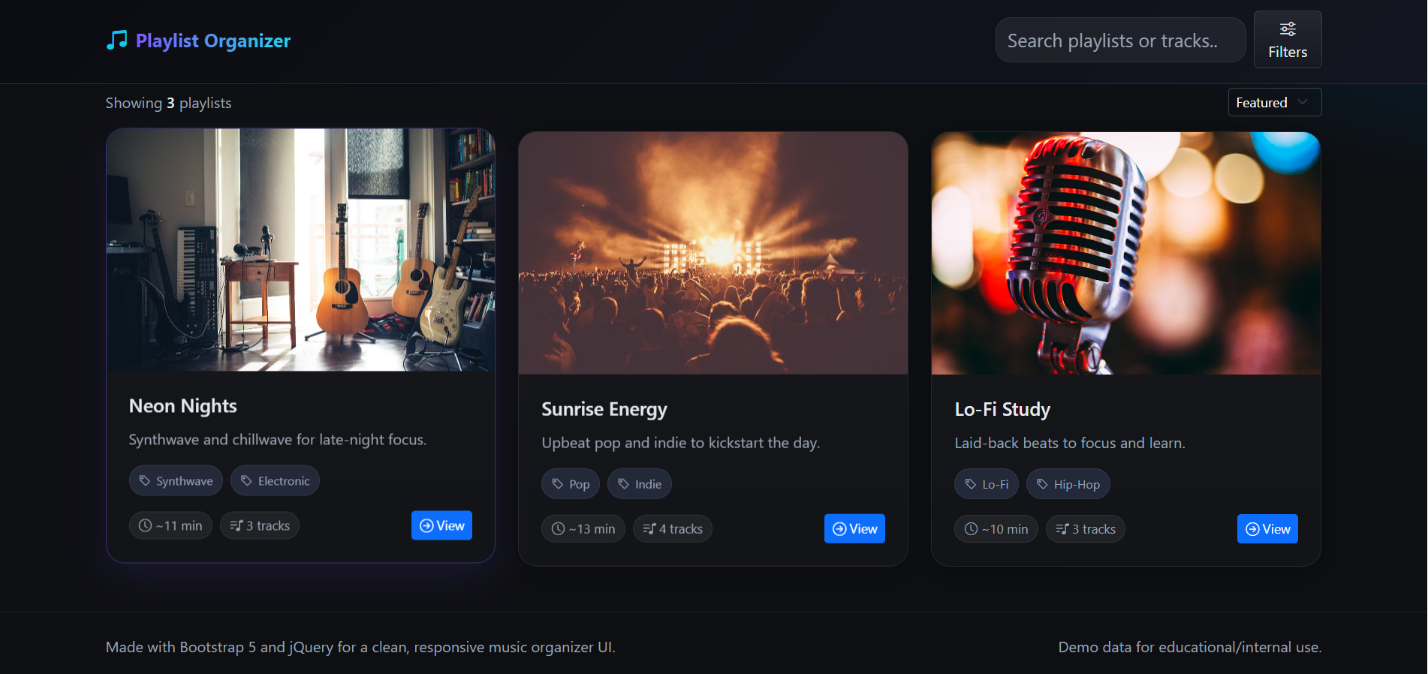
  </script>

</body>

</html>

**13. Screenshots of Final Output**

****

****

**14. Conclusion**

The project successfully delivers a **high-fidelity prototype** of a front-end music organization application. It not only fulfills the technical requirements of responsive design and complex data filtering but also provides an **intuitive user experience** through well-structured navigation and interactive elements like the track playback modal. This foundation provides a strong platform for future development, particularly in expanding to a full-stack application with persistent user data and cloud integration.

**12. References**

* L&T LMS : https://learn.lntedutech.com/Landing/MyCourse