NStream: The Next Entertaining Streaming: Simple, Affordable, and Always Ad-Free

Presented to the
College of Engineering and Architecture
of Mapúa Malayan Colleges Mindanao
Gen. Douglas MacArthur Highway, Davao City

In Partial Fulfillment
of the Requirements for the Course
CPE143L Web Design & Development (Laboratory)

Submitted by:
David Axl Andoy
Ramon Joaquin Fuentes

November 2025

TABLE OF CONTENTS

Table of Contents

Website Title

Nstream

Project Description

NStream is a next-generation streaming service designed to provide affordable, ad-free access to a comprehensive library of movies and series. Built specifically with students and budget-conscious viewers in mind, NStream aims to create a relaxing and personalized entertainment space where quality content meets convenience.

The platform differentiates itself from existing streaming services by offering a low-cost subscription model without compromising content quality, while eliminating disruptive advertisements entirely.

With PC Buy, customers have access to a reliable and well-structured online store tailored for computer parts, making it a valuable tool for tech-savvy shoppers, gamers, developers, and IT professionals.

SWOT Analysis

Strengths

- Affordable Pricing Model: Significantly lower subscription costs compared to mainstream competitors
- Ad-Free Experience: Complete elimination of advertisements enhances user satisfaction
- Target Market Focus: Specifically designed for students and budget-conscious viewers

- Modern Technology Stack: Utilizes current web development technologies (Node.js, Express, HTML5)
- User-Centric Design: Netflix-inspired interface ensures familiarity and ease of use
- Multi-Profile Support: Allows family sharing while maintaining individual preferences
- Comprehensive Feature Set: Includes watchlist, ratings, reviews, and offline downloads
- Clean Architecture: Well-structured database design supports scalability
- Cross-Platform Compatibility: Responsive design works across all devices

Weaknesses

- Limited Content Library: Starting with curated selection rather than extensive catalog
- Mock Subscription System: No real payment integration initially limits monetization
- Small Development Team: Only three team members may limit development speed
- Student Project Status: Academic project may face credibility challenges in real market
- No Original Content: Relies entirely on existing content without unique productions
- Limited Marketing Budget: Academic project constraints limit promotional activities
- Simplified Features: Basic implementation may lack advanced streaming capabilities
- No Mobile App: Web-only platform initially, missing native mobile experience

Opportunities

- Underserved Market: Large gap in affordable, quality streaming services

- Student Market Growth: Expanding student population with limited entertainment budgets
- Regional Expansion: Potential to serve developing markets with cost-sensitive consumers
- Partnership Potential: Collaboration opportunities with educational institutions
- Technology Advancement: Emerging streaming technologies can be integrated
- Content Creator Partnerships: Opportunity to feature independent and local content
- Subscription Tier Expansion: Multiple pricing tiers to capture different market segments
- Educational Content: Potential expansion into educational streaming content

Threats

- Major Competitor Dominance: Netflix, Disney+, Amazon Prime have established market presence
- Content Licensing Costs: Expensive licensing agreements may impact profitability
- Internet Infrastructure: Poor connectivity in target markets may limit accessibility
- Piracy Competition: Free illegal streaming sites provide cost-competitive alternatives
- Economic Downturns: Reduced disposable income may impact subscription renewals
- Regulatory Changes: Streaming regulations and content restrictions may increase compliance costs
- Technology Obsolescence: Rapid technological changes may require constant updates
- Content Creator Direct-to-Consumer: Creators bypassing platforms for direct audience access

Introduction

The digital entertainment landscape has undergone a revolutionary transformation over the past decade, with streaming services becoming the dominant force in how consumers access and enjoy multimedia content. However, this evolution has created a significant disparity in entertainment accessibility, particularly affecting students, young professionals, and budget-conscious families who find themselves priced out of premium streaming experiences.

In the current market ecosystem, major streaming platforms have established oligopolistic pricing structures that often force consumers to choose between multiple expensive subscriptions or settle for ad-interrupted viewing experiences. This creates a substantial barrier to quality entertainment access, especially for demographic segments with limited disposable income. The proliferation of free streaming alternatives, while cost-effective, frequently compromises user experience through excessive advertising, poor video quality, and questionable security protocols.

Problem Statement

Current Market Challenges

- **High subscription costs**: Mainstream streaming services require expensive monthly fees
- **Tiered advertising**: Even paid plans often include disruptive advertisements
- **Security risks**: Free streaming websites pose security threats and offer poor quality
- **Limited accessibility**: Budget-conscious viewers, particularly students, are excluded from premium services

Target Gap

The market lacks an affordable, reliable platform that provides secure, ad-free access to quality entertainment content without financial burden.

Proposed Solution

NStream addresses these challenges by delivering:

- **Streamlined, ad-free streaming platform** at significantly lower cost
- **Accessibility-focused design** with simplicity in mind
- **Curated content library** through low-cost subscription model
- **No hidden fees or advertising interruptions**
- **Clean, dark-themed user interface**
- **Multi-device compatibility** with seamless playback
- **Advanced features** including parental controls, wishlist functionality, and community reviews

User Stories

General Users/Viewers

- User registration and login via email or social media
- Ad-free viewing experience
- Content search and filtering by genre, rating, views or release year
- Multiple profile creation under single account
- Resume watching functionality

Customers/Subscribers

- Flexible payment plans (monthly)
- Secure payment processing (credit/debit cards, digital wallets)
- Subscription history and payment receipt access
- Plan cancellation or upgrade flexibility
- Subscription renewal reminders

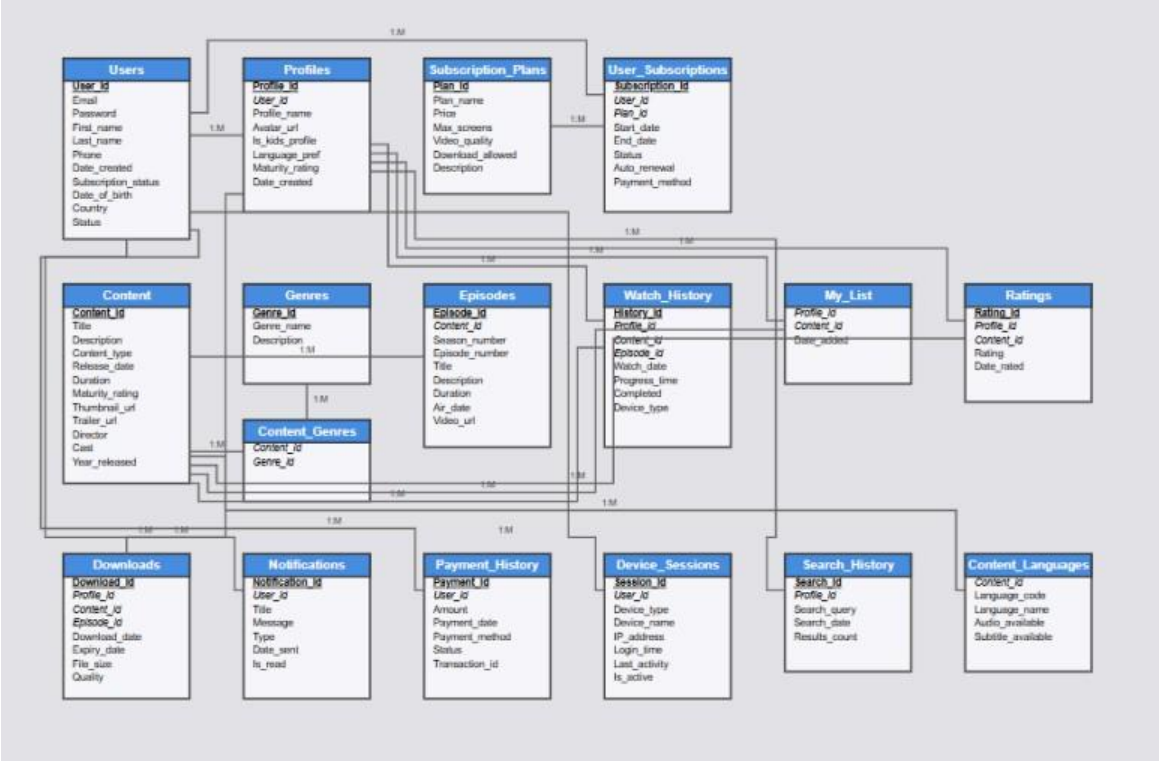
Administrators

- Content upload and management
- User account and subscription management
- Usage statistics monitoring
- Content safety and flagging
- Payment dispute resolution

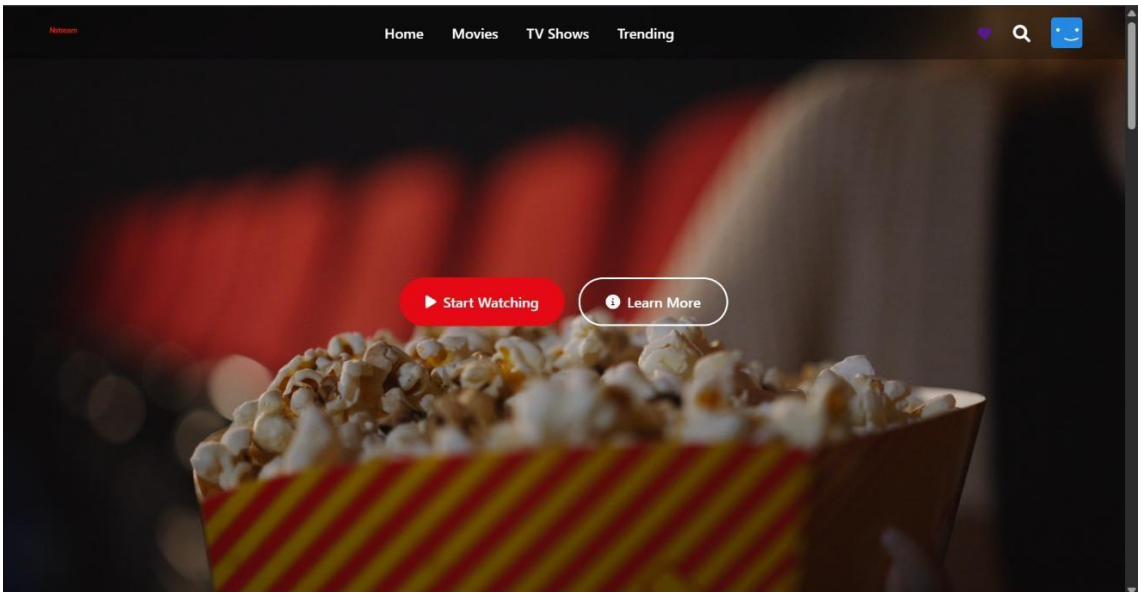
Family/Shared Users

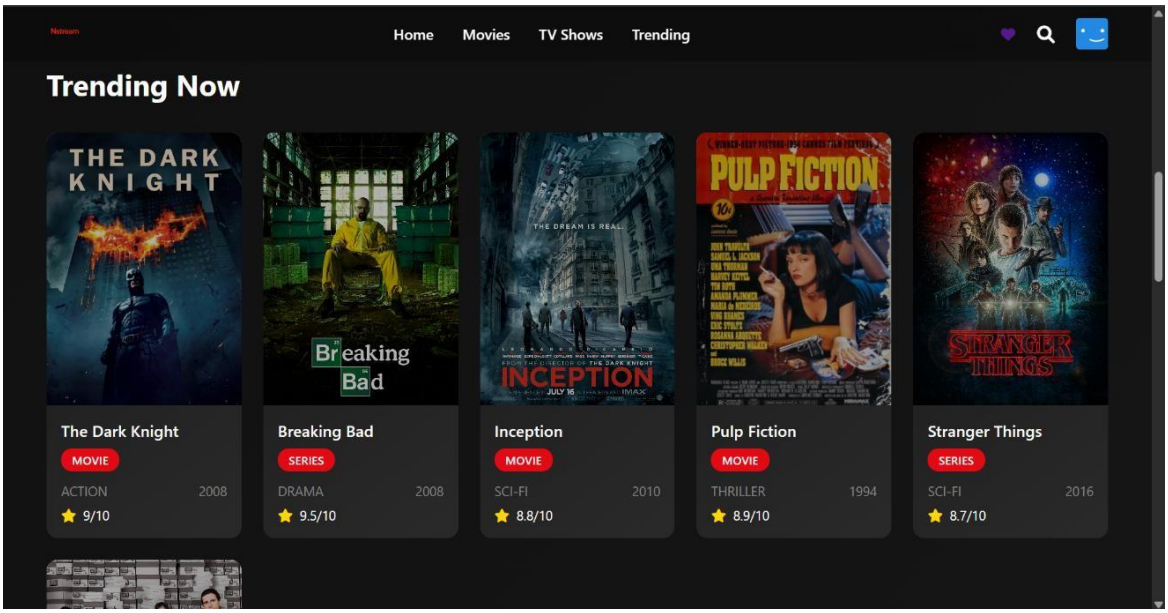
- Individual profile separation for personalized recommendations

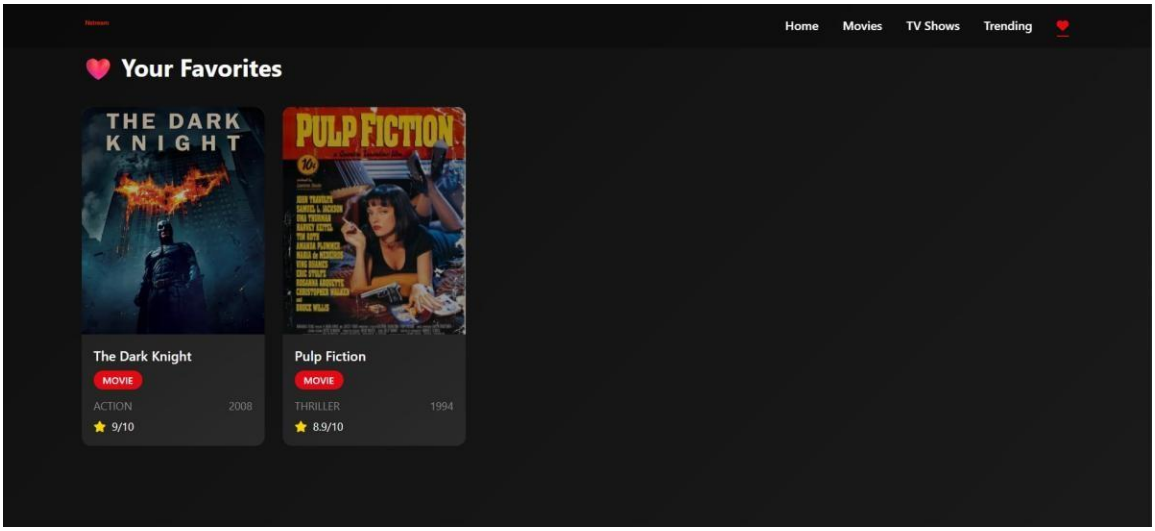
Entity Relationship Diagram

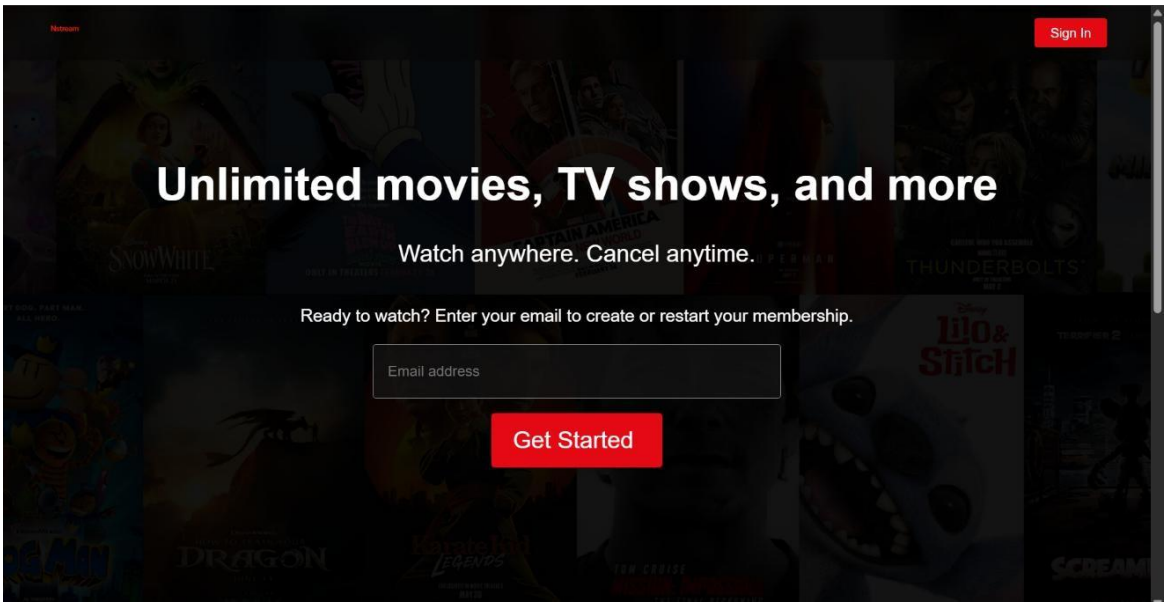


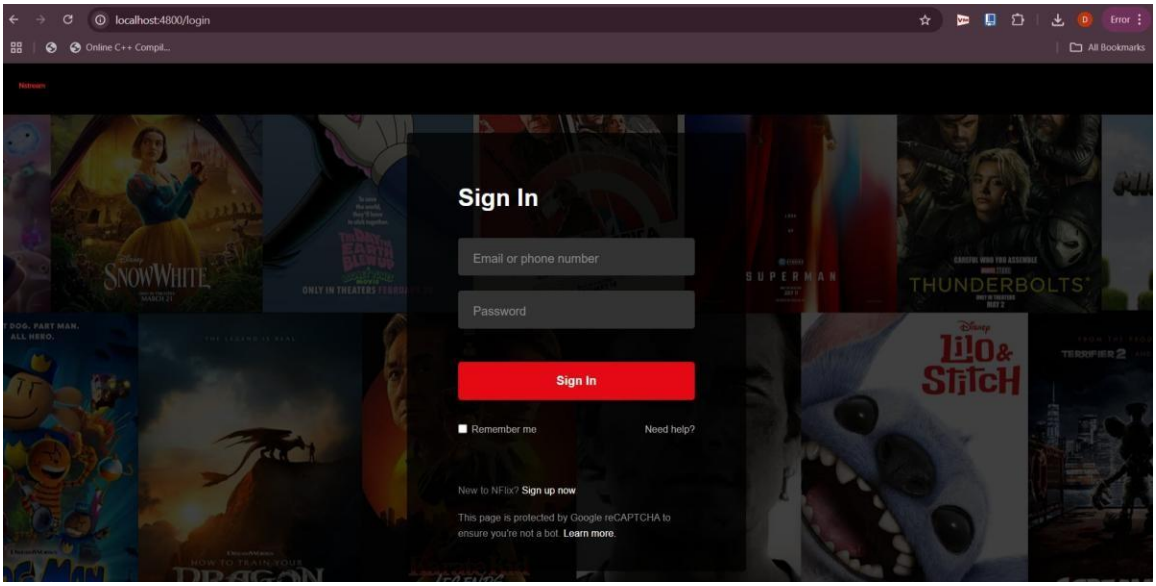
UI Wireframes

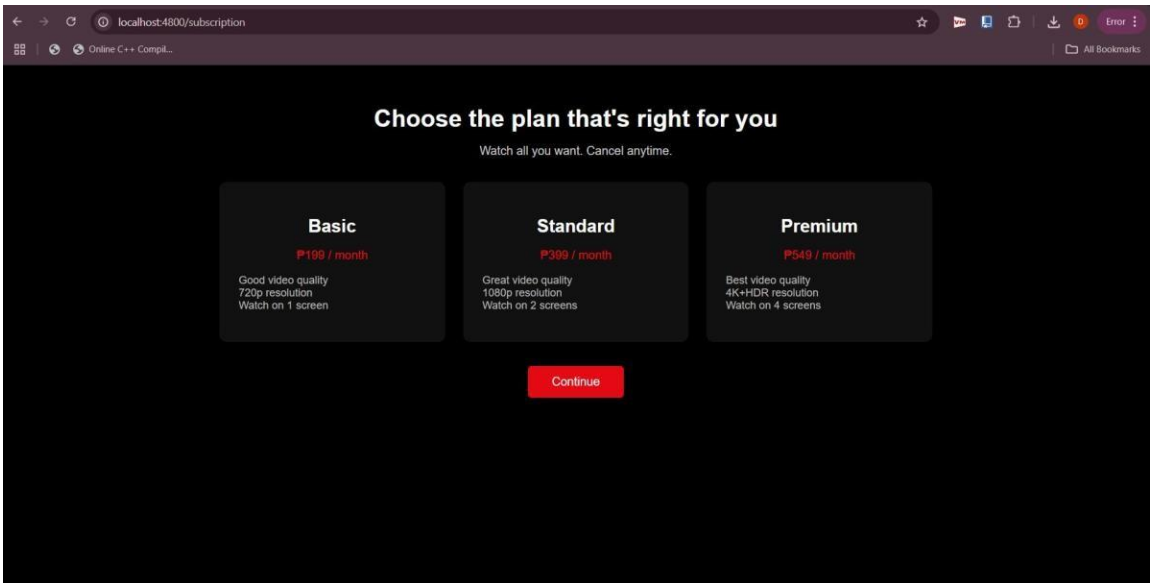


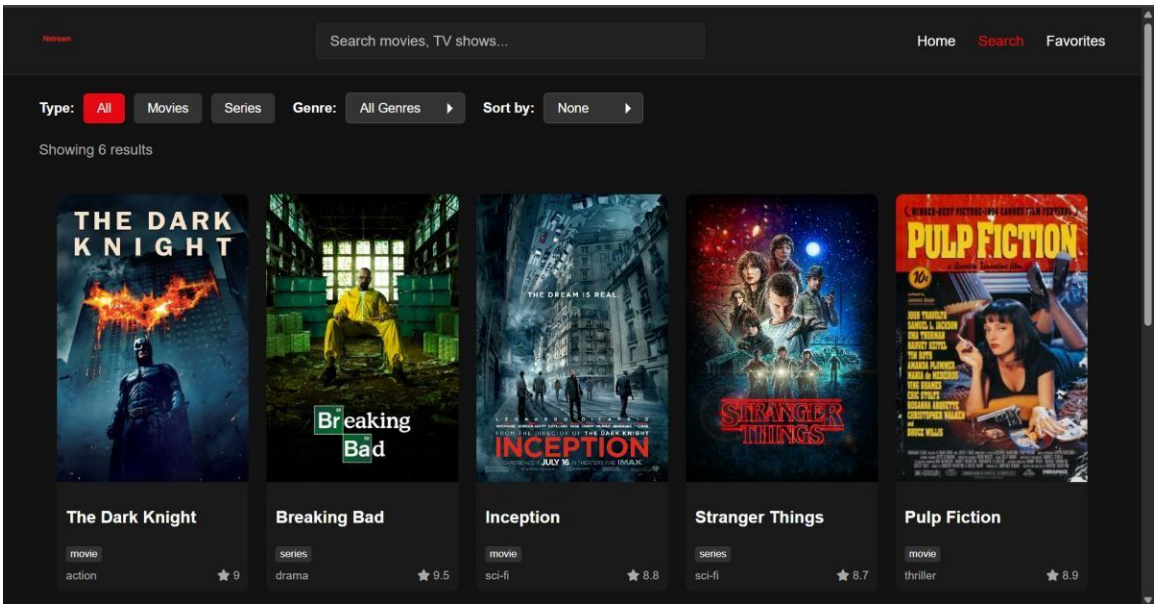

Home
Movies
TV Shows
Trending
Trending Now
The Dark Knight
MOVIE
ACTION
2008
9/10
Breaking Bad
SERIES
DRAMA
2008
9.5/10
Inception
MOVIE
SCI-FI
2010
8.8/10
Pulp Fiction
MOVIE
THRILLER
1994
8.9/10
Stranger Things
SERIES
SCI-FI
2016
8.7/10

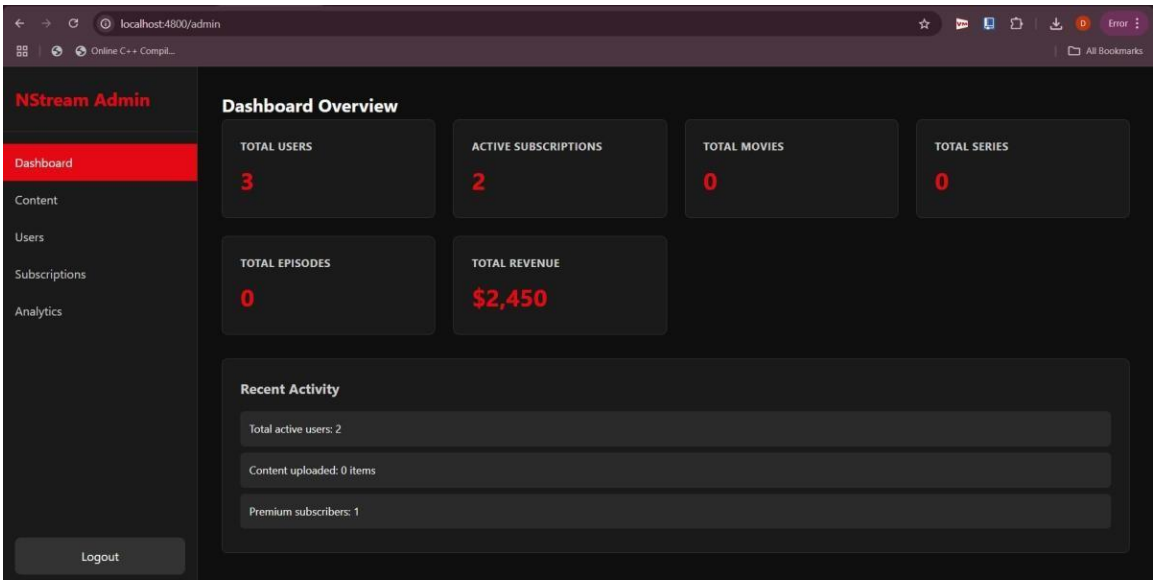

Home
Movies
TV Shows
Trending
Your Favorites
The Dark Knight
MOVIE
ACTION
2008
9/10
Pulp Fiction
MOVIE
THRILLER
1994
8.9/10

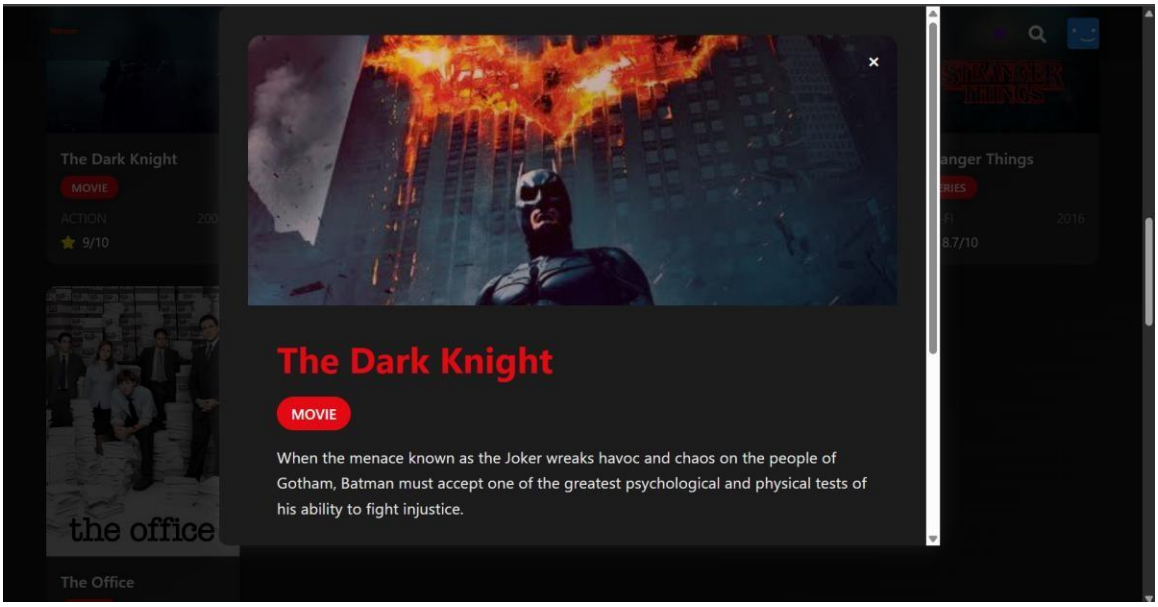

Sign In
Unlimited movies, TV shows, and more
Watch anywhere. Cancel anytime.
Ready to watch? Enter your email to create or restart your membership.
Email address
Get Started

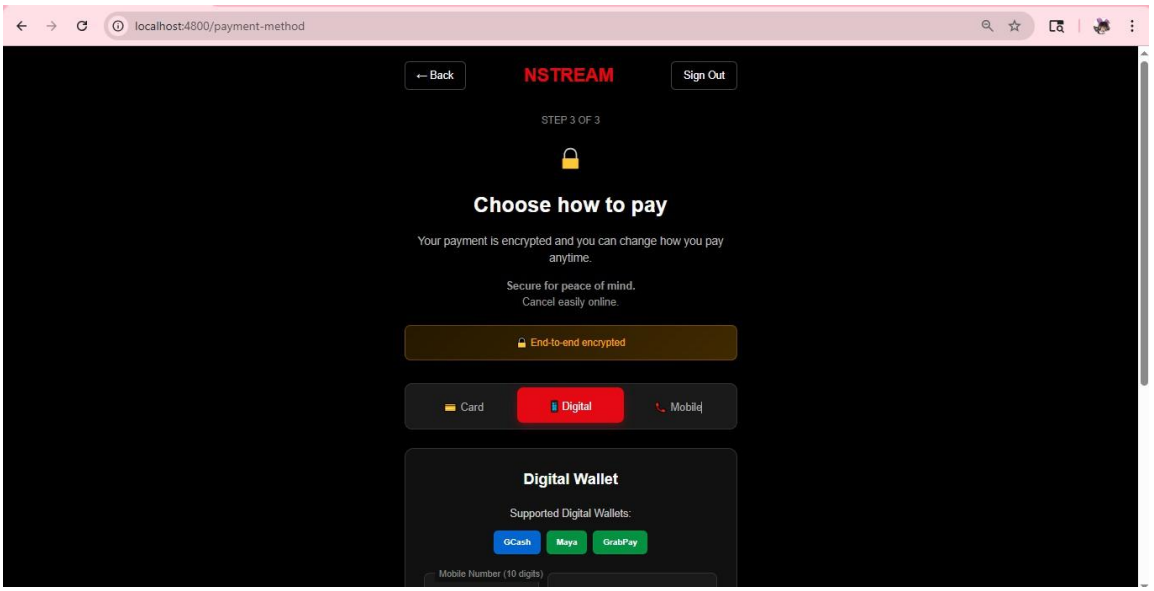

localhost:4800/login
Sign In
Email or phone number
Password
Sign In
Remember me
Need help?
New to Netflix? Sign up now
This page is protected by Google reCAPTCHA to ensure you're not a bot. Learn more.

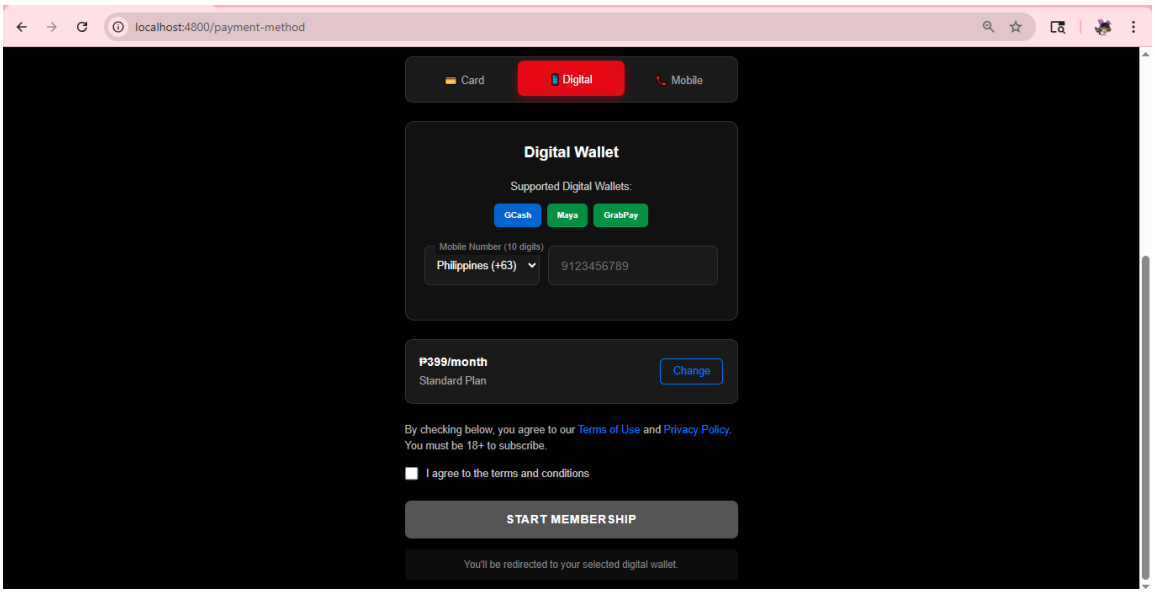

localhost:4800/subscription
Choose the plan that's right for you
Watch all you want. Cancel anytime.
Basic
₱199 / month
Good video quality
720p resolution
Watch on 1 screen
Standard
₱399 / month
Great video quality
1080p resolution
Watch on 2 screens
Premium
₱549 / month
Best video quality
4K+HDR resolution
Watch on 4 screens
Continue


Search movies, TV shows...
Home
Search
Favorites
Type:
All
Movies
Series
Genre:
All Genres
Sort by:
None
Showing 6 results
The Dark Knight
movie
action
9
Breaking Bad
series
drama
9.5
Inception
movie
sci-fi
8.8
Stranger Things
series
sci-fi
8.7
Pulp Fiction
movie
thriller
8.9


localhost:4800/admin
NStream Admin
Dashboard
Content
Users
Subscriptions
Analytics
Logout
Dashboard Overview
TOTAL USERS
3
ACTIVE SUBSCRIPTIONS
2
TOTAL MOVIES
0
TOTAL SERIES
0
TOTAL EPISODES
0
TOTAL REVENUE
$2,450
Recent Activity
Total active users: 2
Content uploaded: 0 items
Premium subscribers: 1


The Dark Knight
MOVIE
When the menace known as the Joker wreaks havoc and chaos on the people of Gotham, Batman must accept one of the greatest psychological and physical tests of his ability to fight injustice.
the office
The Office


localhost:4800/payment-method
← Back
NSTREAM
Sign Out
STEP 3 OF 3
Choose how to pay
Your payment is encrypted and you can change how you pay anytime.
Secure for peace of mind.
Cancel easily online.
End-to-end encrypted
Card
Digital
Mobile
Digital Wallet
Supported Digital Wallets:
GCash
Maya
GrabPay
Mobile Number (10 digits)


localhost:4800/payment-method
Card
Digital
Mobile
Digital Wallet
Supported Digital Wallets:
GCash
Maya
GrabPay
Mobile Number (10 digits)
Philippines (+63)
9123456789
₱399/month
Standard Plan
Change
By checking below, you agree to our Terms of Use and Privacy Policy.
You must be 18+ to subscribe.
I agree to the terms and conditions
START MEMBERSHIP
You'll be redirected to your selected digital wallet.

App Features

- User Registration & Login System - Complete authentication
- Profile Management System - Multi-profile functionality with family sharing
- Content Catalog & Discovery - Advanced search and recommendation engine
- Intelligent Watchlist System - Personal content curation
- Advanced Video Streaming Player - Professional-grade streaming experience
- Responsive Dark-Themed Interface - Modern, accessible design
- Flexible Subscription Management - Transparent pricing and payment options
- Comprehensive Admin Dashboard - Full platform management tools

Technical Implementation:

- Elasticsearch integration for advanced search capabilities
- Dynamic content loading with pagination
- Metadata management system
- Image optimization and lazy loading for performance

Frontend

Core Frontend Stack

- **HTML**: Semantic markup with modern web standards compliance
    - Custom elements and components for reusable UI components
    - Accessibility features with labels and semantic structure
    - optimization with meta tags and structured data
- **CSS**: Advanced styling with modern layout techniques
    - **Flexbox and CSS Grid**: Responsive layout systems
    - **CSS Variables**: Dynamic theming and customization
    - **Media Queries**: Responsive design breakpoints for all device sizes
    - **CSS Animations**: Smooth transitions and micro-interactions
    - **Dark Theme Implementation**: Carefully crafted color palette for eye comfort
- **JavaScript**: Interactive functionality and dynamic behavior
    - **Vanilla JavaScript**: Core functionality without framework dependencies
    - **ES6+ Features**: Modern JavaScript syntax including async/await, modules, and destructuring

Frontend Performance Optimization

- **CDN Integration**: Static asset delivery optimization

Backend Technologies

Server-Side Architecture

- **Node.js**: High-performance JavaScript runtime environment
    - **Event-driven architecture**: Non-blocking I/O operations
    - **Asynchronous programming**: Promise-based async/await patterns
    - **Memory management**: Efficient garbage collection and memory optimization

Requirements

Category	Requirement Description	Status
User Features	Users can browse movies/series without logging in	☐
	Users can search and filter content by title, genre, rating, or year	☐
	Users can select a profile to enter	☐
	Users can create, edit, or delete profiles (name/avatar)	☐
	Users can toggle between multiple user profiles	☐
	Users can favorite (heart) content and view them on a favorites page	☐
	Users can view detailed movie/series pages (info, trailer, etc.)	☐
	Users can simulate video playback (video player page)	☐
Admin Features	Admin can add, edit, or delete content (movies/series)	☐
	Admin can manage user profiles	☐
	Admin can view user activity logs or history (optional)	☐
Authentication	Users can log in and sign up (email/password or social login)	☐

	Session management with localStorage or MongoDB auth	☐
	Route protection for non-authenticated users	☐
Database	Store user profiles, watch history, and favorites in MongoDB/LocalStorage	☐
	Store content data (movies/series info, images, categories)	☐
UI/UX Requirements	"Manage Profiles" UI is responsive and intuitive	☐
	Homepage (mainpage.js) displays featured, trending, and categorized content	☐
	Profile avatars and names are customizable	☐
Performance	Ensure fast content loading and smooth navigation	☐
	Implement lazy loading for images/thumbnails	☐
	Optimize asset delivery (CSS, JS, Images)	☐
Scalability	Design supports adding more genres, categories, or features in future	☐
	Supports multiple concurrent users	☐
Security	Protect user data and profile settings from unauthorized changes	☐
	Admin features hidden from normal users	☐

Entities and Processes:

User (Viewer)

The main end-user who interacts with the system to consume content.

User Activities:

- Select and manage profiles
- Browse, search, and filter movies/series
- View content details and trailers
- Add movies to favorites
- Simulate playback
- Manage account/profile (name, avatar)

Admin

Has elevated privileges for managing the content and user profiles.

Admin Activities:

- Add, edit, or remove content (movies/series)
- Manage user profiles
- View activity or feedback (optional future feature)

MongoDB (Backend Database)

Stores all persistent data for users, content, favorites, and profiles in collections.

MongoDB Collections:

- users: name, avatar, favorites
- content: title, type, genre, rating, releaseDate, posterURL
- favorites: (optional separate collection) links user IDs to content IDs

---

Processes

1.0 Profile Management

Handles profile selection, creation, editing, and deletion.

- Input: User actions (create/edit/delete)
- Output: Updated profile list
- Database: Updates or fetches data from the users collection in MongoDB

2.0 Content Browsing

Displays movie and series lists to the user.

- Input: Page load or user scroll
- Output: List of content items
- Database: Retrieves from the content collection in MongoDB

2.1 Search & Filter

Allows users to search by keyword or filter by genre, rating, etc.

- Input: Search term or filter choice
- Output: Filtered movie list
- Database: Query against the content collection in MongoDB

3.0 Favorite Manager

Adds or removes content from a user's favorites.

- **Input:** User likes/unlikes a movie
- **Output:** Favorites updated
- **Database:** Update the favorites field in the users collection or in a separate favorites collection

---

4.0 Player Interface

Simulates video playback or loads the movie detail page.

- **Input:** User clicks play/view
- **Output:** Movie information and simulated player
- **Database:** Fetches movie metadata from the content collection in MongoDB

5.0 Content Management (Admin)

Admin can add, update, or delete content from the database.

- **Input:** Admin action
- **Output:** Content database changes
- **Database:** Updates performed on the content collection in MongoDB

Use Cases for NStream

Use Case 1: Browse and View Content

- **Actor:** User
- **Description:** The user browses and views available movies and series.
- **Preconditions:** Content must be available in the database.
- **Flow of Events:**
    1. User enters homepage.
    2. Scrolls through featured, trending, and categorized sections.
    3. Clicks on a movie/series to view details.
- **Postconditions:** Movie/series information is displayed.

Use Case 2: Search and Filter

- **Actor:** User
- **Description:** The user searches or filters content based on keywords or categories.
- **Preconditions:** User must be on a content list or homepage.
- **Flow of Events:**
    1. User enters search keyword or selects filter options.
    2. System filters content dynamically.
    3. Results are shown in real-time.
- **Postconditions:** Filtered or searched content list is displayed.

Use Case 3: Add to Favorites

- **Actor:** User
- **Description:** The user adds a movie or series to their personal favorites list.
- **Preconditions:** User must be logged in and have an active profile**.**

- **Flow of Events:**
    1. User clicks the heart icon on a content item.
    2. System adds the content ID to the user's favorites in MongoDB.
- **Postconditions**: Favorite list is updated in the database.

Use Case 4: Simulate Playback

- **Actor:** User
- **Description:** The user opens a content page and simulates viewing the movie/series.
- **Preconditions:** Content must exist in the database.
- **Flow of Events:**
    1. User clicks a movie/series.
    2. System loads the video or description panel.
- **Postconditions:** Playback or information is displayed.

Use Case 5: Manage Profiles

- **Actor:** User
- **Description:** The user creates, edits, or deletes profiles.
- **Preconditions**: User must be logged in.
- **Flow of Events:**
    1. User clicks on "Manage Profiles."
    2. Chooses to add, edit, or delete a profile.
    3. Changes are saved in the MongoDB users collection.
- **Postconditions:** Profile list is updated.

Use Case 6: Admin Manage Content

- **Actor:** Admin
- **Description:** Admin manages movie and series content.
- **Preconditions:** Admin must be logged in.
- **Flow of Events:**
    1. Admin accesses the dashboard.
    2. Views list of content items.
    3. Adds, edits, or deletes entries.
- **Postconditions:** Content collection is updated in MongoDB.

Use Case 7: Admin Manage Users

- **Actor:** Admin
- **Description:** Admin manages user accounts and profiles.
- **Preconditions:** Admin must be logged in.
- **Flow of Events:**
    1. Admin views user list.
    2. Edits or removes a user/profile.
- **Postconditions:** Changes are reflected in the users collection.

Database Design

MongoDB Hierarchical Structure

1. **users Collection**
    - **username:** Unique display name
    - **email:** For authentication
    - **passwordHash:** Securely stored password
    - **avatar:** Chosen profile icon
    - **favorites:** Array of content._id references
    - **isAdmin:** Boolean flag for admin accounts
    - **createdAt:** Timestamp for registration
    -
2. **content Collection**
    - **title:** Name of the content
    - **type:** "movie" or "series"
    - **genre:** Array of genres
    - **description:** Short synopsis
    - **rating:** IMDb-like rating (0.0 - 10.0)
    - **releaseDate:** Release date in ISO format
    - **posterURL**: Link to content poster
    - **studio:** Production or release company
    - **views:** View count for popularity ranking

Database Explanation

NStream uses MongoDB, a NoSQL document-based database, to manage all data related to users, content, and user interactions. Its flexible and scalable design supports the dynamic needs of a modern streaming platform.

1. users Collection

This collection stores all user-related data, including login credentials, profile settings, and personal preferences.

Stores: username, email, passwordHash, avatar, favorites, isAdmin

Each document represents a user account.

The favorites field holds an array of content ObjectIDs the user has saved.

2. content Collection

This is the main collection for storing metadata about movies and series.

Stores: title, type (movie or series), genre, rating, releaseDate, description, posterURL, studio

Used to display lists, search results, and detailed content pages.

Reasons for Using MongoDB

Schema Flexibility: Easily supports future features like reviews, playlists, and watch progress.

Scalability: Works well with growing data like increasing user counts or large movie catalogs.

Speed: Fast read/write operations enable responsive performance for search, filtering, and profile loading.

Work Breakdown Structure for NStream

Planning Phase

- **Defined goal:** Create a streaming site similar to Netflix.
- **Identified main features:** Profiles, favorites, content browsing, admin panel.
- **Tools:** HTML, CSS, JS, Node.js, MongoDB.

System Design Phase

- Frontend broken into modular pages (main, profile, favorites, admin).
- Backend structure using Express and MongoDB.
- Planned schema for users, content, favorites.

UI/UX Design

- Layout based on Netflix interface (dark mode, posters, rows).
- Used responsive design for mobile/tablet/desktop.
- Hover animations, modal transitions, and icon sets added.

Backend Development

- Created API routes for user and content management.
- Set up MongoDB models and connected via Mongoose.
- Authentication with sessions or tokens (if implemented).

Testing and Debugging

- Manual testing across all user flows.
- Checked for broken links, UI bugs, and data sync issues.
- Used dev tools and console logs for debugging.

Deployment and Documentation

- Deployed locally or through platforms like Vercel or Netlify (frontend) and Render or Heroku (backend).

- Created documentation, user manual, and admin guide.

Source Code Sample

Front end

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

  renderContent('moviesGrid', 'movie');

  renderContent('trendingGrid', 'all', 'popularity');

  renderContent('tvshowsGrid', 'series');

  renderFavorites();


  const searchInput = document.getElementById('searchInput');

  if (searchInput) {

    searchInput.addEventListener('keydown', function (e) {

      if (e.key === 'Enter') performSearch(this.value);

    });

  }


  document.querySelectorAll('a[href^="#"]').forEach(anchor => {

    anchor.addEventListener('click', e => {

      e.preventDefault();

      const target =
document.getElementById(anchor.getAttribute('href').substring(1));
```

```
      if (target) target.scrollIntoView({ behavior: 'smooth' });
    });
  });
});

function renderContent(gridId, type, sortBy = 'none') {
  const grid = document.getElementById(gridId);
  if (!grid) return;

  grid.innerHTML = '';

  let data = filteredData.filter(item => type === 'all' || item.type === type);
  if (sortBy !== 'none') {
    data.sort((a, b) => {
      switch (sortBy) {
        case 'popularity': return b.popularity - a.popularity;
        case 'views': return b.views - a.views;
        case 'rating': return b.rating - a.rating;
        case 'year': return b.year - a.year;
        default: return 0;
      }
    });
  }

  data.forEach(item => {
    const card = createContentCard(item);
    grid.appendChild(card);
  });
```

```
}

function createContentCard(item) {
  const card = document.createElement("div");
  card.className = "movie-card";
  card.dataset.itemId = item.id;
  card.addEventListener("click", () => openModal(item));

  const displayType = item.type === "series" ? "SERIES" : item.type.toUpperCase();

  card.innerHTML = `
    <div class="movie-poster">
      <img src="${item.poster || 'assets/images/default-poster.png'}"
alt="${item.title} Poster" onerror="this.src='assets/images/default-poster.png'"
/>
    </div>
    <div class="movie-info">
      <h3 class="movie-title">${item.title}</h3>
      <div class="movie-type">${displayType}</div>
      <div class="movie-info-content">
        <span class="movie-genre">${item.genre.toUpperCase()}</span>
        <span>${item.year}</span>
      </div>
      <div class="movie-rating">
        <span class="rating-stars"><i class="fas fa-star star"></i></span>
        <span>${item.rating}/10</span>
      </div>
    </div>
  `;
```

```
  return card;
}

function renderFavorites() {
  const profile = getCurrentProfile();
  const grid = document.getElementById("favoritesGrid");
  if (!grid) return;

  if (!profile) {
    grid.innerHTML = "<p style='text-align: center;'>Please select a profile to view favorites.</p>";
    return;
  }

  const favorites = getFavorites();
  grid.innerHTML = "";

  if (!favorites.length) {
    grid.innerHTML = "<p style='text-align: center;'>No favorites added yet.</p>";
    return;
  }

  favorites.forEach(item => {
    const card = createContentCard(item);
    grid.appendChild(card);
  });
}
```

```
function toggleSearch() {
  const searchOverlay = document.getElementById('searchOverlay');
  searchOverlay.style.display = (searchOverlay.style.display === 'flex') ? 'none'
: 'flex';
}

function filterContent(genre) {
  currentGenreFilter = genre;
  document.querySelectorAll('.category-btn').forEach(btn                          =>
btn.classList.remove('active'));
  document.querySelector(`.category-
btn[onclick="filterContent('${genre}')"]`).classList.add('active');
  applyFilters();
}

function  performSearch(query)  {
  currentSearchTerm = query.trim().toLowerCase();
  applyFilters();
  document.getElementById('movies').scrollIntoView({ behavior: 'smooth' });
  toggleSearch();
}

function applyFilters() {
  filteredData = contentData.filter(item => {
    const matchesGenre = currentGenreFilter === 'all' || item.genre.toLowerCase()
=== currentGenreFilter.toLowerCase();
    const matchesSearch = currentSearchTerm === '' ||
      item.title.toLowerCase().includes(currentSearchTerm) ||
```

```
      item.genre.toLowerCase().includes(currentSearchTerm) ||
      item.description.toLowerCase().includes(currentSearchTerm);
    return matchesGenre && matchesSearch;
  });

  renderContent('moviesGrid', 'movie');
  renderContent('trendingGrid', 'all', 'popularity');
  renderContent('tvshowsGrid', 'series');
}

function openModal(item) {
  const modal = document.getElementById('contentModal');
  const modalTitle = document.getElementById('modalTitle');
  const modalType = document.getElementById('modalType');
  const modalDescription = document.getElementById('modalDescription');
  const episodesSection = document.getElementById('episodesSection');
  const episodesList = document.getElementById('episodesList');
  const likeBtn = document.getElementById('likeBtn');
  const likeText = document.getElementById('likeText');
  const typeDisplay = item.type === 'series' ? 'SERIES' : item.type.toUpperCase();

  // Set modal content
  modalTitle.textContent = item.title;
  modalType.textContent = typeDisplay;
  modalDescription.textContent = item.description;
  document.getElementById('modalGenre').textContent =
item.genre.charAt(0).toUpperCase() + item.genre.slice(1);
  document.getElementById('modalStudio').textContent = item.studio;
```

```
  document.getElementById('modalReleaseDate').textContent = item.releaseDate;

  document.getElementById('modalViews').textContent = item.viewsFormatted;

  document.getElementById('modalPopularity').textContent = `${item.popularity}%`;

  document.getElementById('modalRating').textContent = `${item.rating}/10`;

  document.getElementById('modalParentalGuidance').textContent = item.parentalGuidance;


  // Update modal header with poster

  const modalHeader = document.querySelector('.modal-header');

  modalHeader.innerHTML = `

    <img src="${item.poster || 'assets/images/default-poster.png'}" alt="${item.title} Poster" class="modal-poster" onerror="this.src='assets/images/default-poster.png'" />

    <button class="modal-close" onclick="closeModal()">×</button>

  `;


  // Handle series status

  const statusRow = document.getElementById('modalStatusRow');

  if (item.type === 'series' && item.status) {

    statusRow.style.display = 'flex';

    document.getElementById('modalStatus').textContent = item.status.charAt(0).toUpperCase() + item.status.slice(1);

  } else {

    statusRow.style.display = 'none';

  }


  // Handle episodes for series

  if (item.type === 'series' && item.episodes) {

    episodesSection.style.display = 'block';
```

```
    episodesList.innerHTML = item.episodes.map(episode => `
      <div class="episode-card">
        <div class="episode-content">
          <div class="episode-video">
            <div class="episode-thumbnail">
              <i class="fas fa-play-circle"></i>
              <div class="episode-duration">${episode.duration}</div>
            </div>
          </div>
          <div class="episode-details">
            <div    class="episode-title">S${episode.season}E${episode.episode}:
${episode.title}</div>
            <div class="episode-info">
              <span>Released: ${episode.releaseDate}</span>
              <div class="episode-stats">
                <span><i class="fas fa-star star"></i> ${episode.rating}</span>
                <span><i                 class="fas                 fa-eye"></i>
${episode.viewsFormatted}</span>
                <span><i class="fas fa-fire"></i> ${episode.popularity}%</span>
              </div>
            </div>
            <div class="episode-description">${episode.description}</div>
            <button  class="episode-watch-btn"  onclick="watchEpisode(${item.id},
${episode.id})">
              <i class="fas fa-play"></i> Watch Episode
            </button>
          </div>
        </div>
      </div>
```

```
    `).join('');
  } else {
    episodesSection.style.display = 'none';
  }

  // Check if item is in favorites and set button state
  const favorites = getFavorites();
  const isFavorited = favorites.some(fav => fav.id === item.id);
  if (isFavorited) {
    likeBtn.classList.add('active');
    likeText.textContent = 'Liked';
  } else {
    likeBtn.classList.remove('active');
    likeText.textContent = 'Like';
  }

  modal.style.display = 'block';
}

function getCurrentProfile() {
  return JSON.parse(sessionStorage.getItem("currentProfile"));
}

function getFavorites() {
  const profile = getCurrentProfile();
  if (!profile) return [];
  return JSON.parse(sessionStorage.getItem(`favorites_${profile.name}`)) || [];
}
```

```
function saveFavorites(favorites) {
  const profile = getCurrentProfile();
  if (!profile) return;
  sessionStorage.setItem(`favorites_${profile.name}`,
JSON.stringify(favorites));
}

function toggleLike() {
  const btn = document.getElementById('likeBtn');
  const text = document.getElementById('likeText');
  const modalTitle = document.getElementById('modalTitle').textContent;
  const content = contentData.find(item => item.title === modalTitle);

  if (!content) return;

  const profile = getCurrentProfile();
  if (!profile) {
    alert('Please select a profile to like content.');
    return;
  }

  let favorites = getFavorites();

  if (btn.classList.contains('active')) {
    // Remove from favorites
    favorites = favorites.filter(fav => fav.id !== content.id);
    btn.classList.remove('active');
```

```
    text.textContent = 'Like';
  } else {
    // Add to favorites
    favorites.push(content);
    btn.classList.add('active');
    text.textContent = 'Liked';
  }

  saveFavorites(favorites);
  renderFavorites();
}

function toggleFavorite(movie) {
  const favorites = getFavorites();
  const index = favorites.findIndex(fav => fav.id === movie.id);

  if (index !== -1) {
    // Already in favorites - remove
    favorites.splice(index, 1);
  } else {
    // Add to favorites
    favorites.push(movie);
  }

  saveFavorites(favorites);
  renderFavorites();
  updateLikeButton(movie.id);
}
```

```
function updateLikeButton(movieId) {
  const button = document.querySelector(`.like-btn[data-id="${movieId}"]`);
  if (!button) return;

  const favorites = getFavorites();
  const isFavorite = favorites.some(m => m.id === movieId);
  button.classList.toggle("liked", isFavorite);
}

function resetUserActions() {
  const likeBtn = document.getElementById('likeBtn');
  const likeText = document.getElementById('likeText');
  likeBtn.classList.remove('active');
  likeText.textContent = 'Like';
}

function watchNow() {
  const modalTitle = document.getElementById('modalTitle').textContent;
  const content = contentData.find(item => item.title === modalTitle);

  if (!content) {
    alert('Content not found');
    return;
  }

  if (!content.videoFile) {
    alert('Video file not available');
```

```
    return;
  }

  // Store the content data and video file path for the watch page
  sessionStorage.setItem('currentWatchContent', JSON.stringify(content));
  sessionStorage.setItem('currentVideoFile', content.videoFile);

  // Navigate to watch page
  window.location.href = 'watchpage.html';
}

function watchEpisode(seriesId, episodeId) {
  const series = contentData.find(item => item.id === seriesId);
  if (!series || !series.episodes) {
    alert('Series not found');
    return;
  }

  const episode = series.episodes.find(ep => ep.id === episodeId);
  if (!episode) {
    alert('Episode not found');
    return;
  }

  if (!episode.videoFile) {
    alert('Episode video file not available');
    return;
  }
```

```
  // Create episode content object for watch page
  const episodeContent = {
    ...episode,
    seriesTitle: series.title,
    seriesId: series.id,
    type: 'episode'
  };

  // Store the episode data and video file path for the watch page
  sessionStorage.setItem('currentWatchContent', JSON.stringify(episodeContent));
  sessionStorage.setItem('currentVideoFile', episode.videoFile);

  // Navigate to watch page
  window.location.href = 'watchpage.html';
}

function closeModal() {
  document.getElementById('contentModal').style.display = 'none';
}

function openLearnMore() {
  let modal = document.getElementById("learnMoreModal");
  if (!modal) {
    modal = document.createElement("div");
    modal.className = "modal";
    modal.id = "learnMoreModal";
    modal.innerHTML = `
```

```
      <div class="modal-content">
        <button class="modal-close" onclick="closeLearnMore()">×</button>
        <h2>About NStream</h2>
        <p>NStream is your ultimate entertainment hub for streaming blockbuster movies, trending series, and exclusive originals—all in one platform with zero ads and unlimited access.</p>
      </div>
    `;
    document.body.appendChild(modal);
  }
  modal.style.display = "flex";
}

function closeLearnMore() {
  const modal = document.getElementById("learnMoreModal");
  if (modal) modal.style.display = "none";
}

function getProfiles() {
  return JSON.parse(sessionStorage.getItem("profiles")) || [];
}

function setCurrentProfile(profile) {
  sessionStorage.setItem("currentProfile", JSON.stringify(profile));
  updateNavbarProfile(profile);
}

function openChangeProfileModal() {
  const modal = document.getElementById("changeProfileModal");
```

```
  const list = document.getElementById("changeProfileList");
  const profiles = getProfiles();

  list.innerHTML = "";

  profiles.forEach((profile, index) => {
    const div = document.createElement("div");
    div.className = "profile-card";
    div.innerHTML = `
      <img src="${profile.avatar}" alt="${profile.name}" />
      <p>${profile.name}</p>
    `;
    div.onclick = () => {
      setCurrentProfile(profile);
      closeChangeProfileModal();
    };
    list.appendChild(div);
  });

  modal.style.display = "flex";
}

function closeChangeProfileModal() {
  document.getElementById("changeProfileModal").style.display = "none";
}

function updateNavbarProfile(profile) {
  const avatarEl = document.querySelector(".profile-avatar");
```

```
  if (avatarEl && profile) {
    avatarEl.src = profile.avatar;
    avatarEl.alt = profile.name;
  }
}

function openHelpCenter() {
  document.getElementById("helpCenterModal").style.display = "flex";
}

function closeHelpCenter() {
  document.getElementById("helpCenterModal").style.display = "none";
}

function toggleProfileMenu() {
  const dropdown = document.getElementById("profileDropdown");
  dropdown.classList.toggle("hidden");
}

window.onload = () => {
  const profile = getCurrentProfile();
  if (profile) {
    updateNavbarProfile(profile);
  }
};

// Close modal when clicking outside
window.onclick = function(event) {
```

```
  const modal = document.getElementById('contentModal');
  if (event.target === modal) {
    closeModal();
  }
}

// Close modal with Escape key
document.addEventListener('keydown', function(event) {
  if (event.key === 'Escape') {
    closeModal();
  }
});

function openInfoModal() {
  const infoModal = document.getElementById('infoModal');
  if (infoModal) {
    infoModal.style.display = 'flex';
  }
}

function closeInfoModal() {
  const infoModal = document.getElementById('infoModal');
  if (infoModal) {
    infoModal.style.display = 'none';
  }
}
function goToWatchPage(item) {
  localStorage.setItem("watchingItem", JSON.stringify(item));
```

```
  window.location.href = 'watch.html';
}
// STEP 2: Profile Management Functions

// Get all stored profiles
function getProfiles() {
  return JSON.parse(sessionStorage.getItem("profiles")) || [];
}

// Set the current profile (when switching)
function setCurrentProfile(profile) {
  sessionStorage.setItem("currentProfile", JSON.stringify(profile));
  updateNavbarProfile(profile); // Step 3: Update navbar immediately
  location.reload(); // Optional: Refresh page to apply changes
}

// Get the current active profile
function getCurrentProfile() {
  return JSON.parse(sessionStorage.getItem("currentProfile"));
}

// STEP 3: Update navbar avatar dynamically
function updateNavbarProfile(profile) {
  const avatarEl = document.querySelector(".profile-avatar");
  if (avatarEl && profile) {
    avatarEl.src = profile.avatar || "assets/images/default-avatar.png";
    avatarEl.alt = profile.name;
  }
```

```
}


// Transfer Profile Modal Logic
function openChangeProfileModal() {
  const modal = document.getElementById("changeProfileModal");
  const list = document.getElementById("changeProfileList");
  const profiles = getProfiles();

  list.innerHTML = "";

  profiles.forEach(profile => {
    const div = document.createElement("div");
    div.className = "profile-card";
    div.innerHTML = `
      <img src="${profile.avatar}" alt="${profile.name}" />
      <p>${profile.name}</p>
    `;
    div.onclick = () => {
      setCurrentProfile(profile);      // store to sessionStorage
      closeChangeProfileModal();      // close modal
    };
    list.appendChild(div);
  });

  modal.style.display = "flex";
}

function closeChangeProfileModal() {
```

```
  const modal = document.getElementById("changeProfileModal");
  if (modal) modal.style.display = "none";
}

// Show dropdown menu for profile actions
function toggleProfileMenu() {
  const dropdown = document.getElementById("profileDropdown");
  if (dropdown) dropdown.classList.toggle("hidden");
}

// On page load, auto-update navbar if profile exists
window.onload = () => {
  const profile = getCurrentProfile();
  if (profile) {
    updateNavbarProfile(profile);
  }
};
window.onload = () => {
  const profile = getCurrentProfile();
  if (!profile) {
    window.location.href = "profiles.html"; // or your profile select page
  } else {
    updateNavbarProfile(profile);
  }
};
function getProfiles() {
  // Pull from localStorage — these are the ones created in manage.js
  return JSON.parse(localStorage.getItem("profiles")) || [];
```

```
}

function setCurrentProfile(profile) {
  // Save selected profile into sessionStorage (active session only)
  sessionStorage.setItem("currentProfile", JSON.stringify(profile));
  updateNavbarProfile(profile);
  location.reload(); // Optional: Refresh to re-render based on new profile
}

function openChangeProfileModal() {
  const modal = document.getElementById("changeProfileModal");
  const list = document.getElementById("changeProfileList");
  const profiles = getProfiles(); // ← pulling from localStorage

  list.innerHTML = "";

  profiles.forEach(profile => {
    const div = document.createElement("div");
    div.className = "profile-card";
    div.innerHTML = `
      <img src="${profile.avatar}" alt="${profile.name}" />
      <p>${profile.name}</p>
    `;
    div.onclick = () => {
      setCurrentProfile(profile); // ← store to sessionStorage
      closeChangeProfileModal(); // ← close modal
    };
    list.appendChild(div);
```

```
  });


  modal.style.display = "flex";
}


function closeChangeProfileModal() {
  document.getElementById("changeProfileModal").style.display = "none";
}
function updateNavbarProfile(profile) {
  const avatarEl = document.querySelector(".profile-avatar");
  if (avatarEl && profile) {
    avatarEl.src = profile.avatar;
    avatarEl.alt = profile.name;
  }
}


window.onload = () => {
  const profile = JSON.parse(sessionStorage.getItem("currentProfile"));
  if (profile) updateNavbarProfile(profile);
};
```

Backend

DbConnection:

```
require('dotenv').config();
const mongoose = require('mongoose');
```

```
const connectDB = async () => {
  try {
    console.log('🔄 Connecting to MongoDB...');

    await mongoose.connect(process.env.MONGO_URI);

    console.log('✅ MongoDB Connected Successfully');
    console.log(`- Host: ${mongoose.connection.host}`);
    console.log(`- Database: ${mongoose.connection.name}`);

    // Connection events
    mongoose.connection.on('connected', () => {
      console.log('🟢 Mongoose connected to DB');
    });

    mongoose.connection.on('error', (err) => {
      console.error('🔴 Mongoose connection error:', err);
    });

    mongoose.connection.on('disconnected', () => {
      console.log('🟠 Mongoose disconnected');
    });

    // Graceful shutdown
    process.on('SIGINT', async () => {
      await mongoose.connection.close();
      console.log('🔒 Mongoose connection closed');
```

```
      process.exit(0);
    });

  } catch (error) {
    console.error('+ MongoDB Connection Error:', error.message);
    console.log('\n  Troubleshooting Steps:');
    console.log('1. Check MONGO_URI in .env file');
    console.log('2. Verify IP whitelisting in MongoDB Atlas');
    console.log('3. Check internet connection');
    process.exit(1);
  }
};

module.exports = connectDB;
```

R2Connection:

```
const { S3Client, ListObjectsV2Command } = require('@aws-sdk/client-s3');

const client = new S3Client({
  region: "auto",
  endpoint: process.env.R2_ENDPOINT,
  credentials: {
    accessKeyId: process.env.R2_ACCESS_KEY_ID.trim(),
    secretAccessKey: process.env.R2_SECRET_ACCESS_KEY.trim()
  }
});

async function connectR2() {
```

```
  try {
    console.log('□ Connecting to Cloudflare R2...');
    await client.send(new ListObjectsV2Command({
      Bucket: process.env.R2_BUCKET_NAME,
      MaxKeys: 1
    }));
    console.log('☑ Cloudflare R2 connected successfully');
    return client;
  } catch (error) {
    console.error('☒ R2 Connection Error:', error.message);
    throw error;
  }
}

module.exports = connectR2;
module.exports.client = client; // Also export the client directly
```

index.js:

```
require('dotenv').config();
const express = require('express');
const path = require('path');
const mongoose = require('mongoose');
const session = require('express-session');
const MongoStore = require('connect-mongo');
const bcrypt = require('bcryptjs');
const connectDB = require('./dBConnection');
const connectR2 = require('./r2Connection');
```

```
// Initialize app
const app = express();
const PORT = process.env.PORT || 4800;

// Import routes
const uploadRoutes = require('./public/routes/uploads');

const pagesRoutes = require('./public/routes/pages');

// Middleware
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// FIXED: Static file serving - serve both root directory and public folder
app.use(express.static(path.join(__dirname, 'public')));
app.use(express.static(__dirname)); // This serves files from root directory

// Session middleware
app.use(session({
  secret: process.env.SESSION_SECRET,
  resave: false,
  saveUninitialized: false,
  store: MongoStore.create({
    mongoUrl: process.env.MONGO_URI,
    ttl: 14 * 24 * 60 * 60 // 14 days
  }),
  cookie: {
```

```
    secure: process.env.NODE_ENV === 'production',
    maxAge: 1000 * 60 * 60 * 24 * 14 // 14 days
  }
}));

// User model
const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  isAdmin: { type: Boolean, default: false },
  createdAt: { type: Date, default: Date.now }
});
const User = mongoose.model('User', userSchema);

// Content model (for admin dashboard)
const contentSchema = new mongoose.Schema({
  title: String,
  type: String,
  // Add other content fields as needed
});
const Content = mongoose.model('Content', contentSchema);

// Routes
app.use('/api/upload', uploadRoutes);
app.use('/', pagesRoutes);

// Auth endpoints
```

```
app.post('/api/register', async (req, res) => {
  try {
    const { name, email, password, confirmPassword } = req.body;

    if (password !== confirmPassword) {
      return res.status(400).json({
        success: false,
        message: 'Passwords do not match'
      });
    }

    const existingUser = await User.findOne({ email });
    if (existingUser) {
      return res.status(400).json({
        success: false,
        message: 'Email already in use'
      });
    }

    const hashedPassword = await bcrypt.hash(password, 10);
    const newUser = await User.create({ name, email, password: hashedPassword });

    res.json({
      success: true,
      message: 'Registration successful!',
      user: { id: newUser._id, name: newUser.name }
    });
```

```
  } catch (error) {
    console.error('Registration error:', error);
    res.status(500).json({
      success: false,
      message: 'Registration failed. Please try again.'
    });
  }
});

app.post('/api/login', async (req, res) => {
  try {
    const { email, password } = req.body;
    const user = await User.findOne({ email });

    if (!user || !(await bcrypt.compare(password, user.password))) {
      return res.status(401).json({
        success: false,
        message: 'Invalid email or password'
      });
    }

    req.session.user = { id: user._id, name: user.name, email: user.email };
   res.json({ success: true, message: 'Login successful!', user: req.session.user
});

  } catch (error) {
    console.error('Login error:', error);
    res.status(500).json({
```

```
      success: false,
      message: 'Login failed. Please try again.'
    });
  }
});

app.post('/api/logout', (req, res) => {
  req.session.destroy(err => {
    if (err) return res.status(500).json({ success: false, message: 'Logout failed' });
    res.clearCookie('connect.sid');
    res.json({ success: true, message: 'Logged out successfully' });
  });
});

// Admin endpoints
app.get('/api/admin/stats', async (req, res) => {
  try {
    if (!req.session.user?.isAdmin) {
      return res.status(403).json({ success: false, message: 'Unauthorized' });
    }

    const [userCount, subCount, movieCount, seriesCount] = await Promise.all([
      User.countDocuments(),
      User.countDocuments({ subscription: { $ne: 'Free' } }),
      Content.countDocuments({ type: 'Movie' }),
      Content.countDocuments({ type: 'Series' })
    ]);
```

```
      res.json({ success: true, stats: { userCount, subCount, movieCount, seriesCount } });
  } catch (error) {
    console.error('Admin stats error:', error);
    res.status(500).json({ success: false, message: 'Failed to load stats' });
  }
});

// Error handling
app.use((err, req, res, next) => {
  console.error('Error:', err);
  res.status(500).json({ success: false, message: 'Server error' });
});

// Start server
const startServer = async () => {
  try {
    console.log('Connecting to databases...');
    await connectDB();
    await connectR2();

    app.listen(PORT, () => {
      console.log(`Server running on http://localhost:${PORT}`);
    });
  } catch (err) {
    console.error('Server startup failed:', err);
    process.exit(1);
```

```
  }
};
```

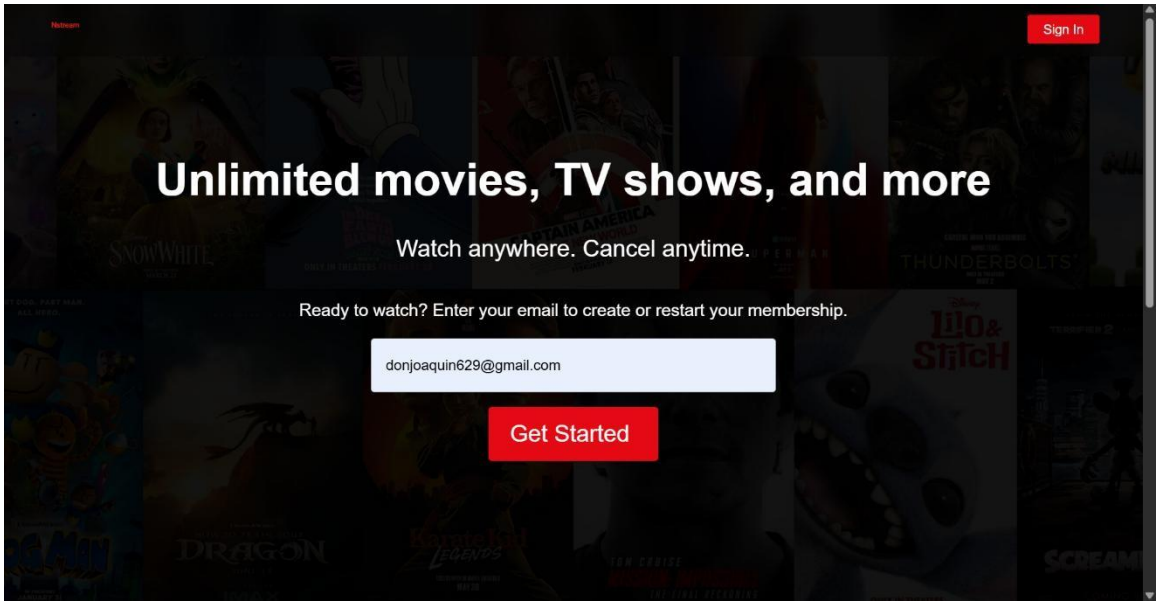
```
// Enhanced r2Connection.js with better error handling
const { S3Client, ListObjectsV2Command } = require('@aws-sdk/client-s3');

const client = new S3Client({
  region: "auto",
  endpoint: process.env.R2_ENDPOINT,
  credentials: {
    accessKeyId: process.env.R2_ACCESS_KEY_ID.trim(),
    secretAccessKey: process.env.R2_SECRET_ACCESS_KEY.trim()
  }
});

module.exports = connectR2;
module.exports.client = client;
startServer();
```

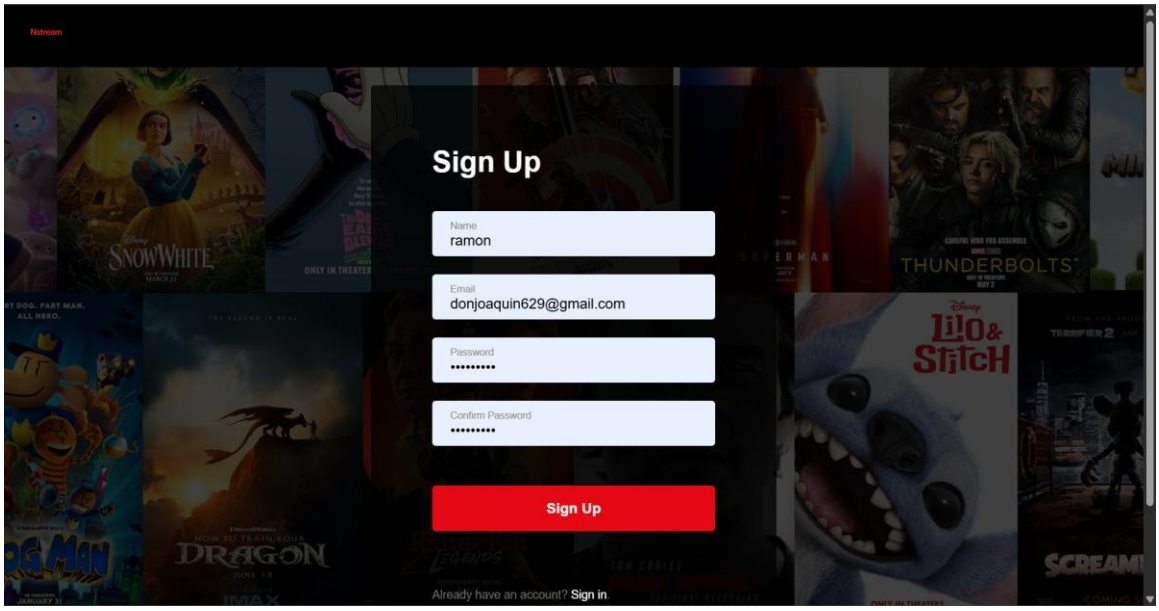
User Manual

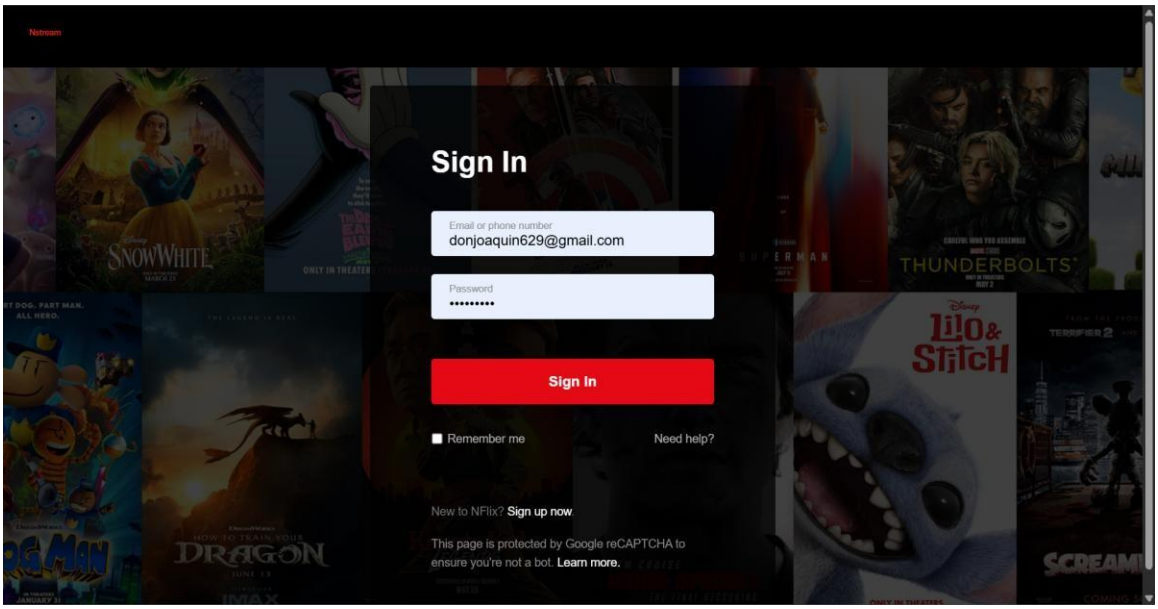
User

1. Accessing the Platform

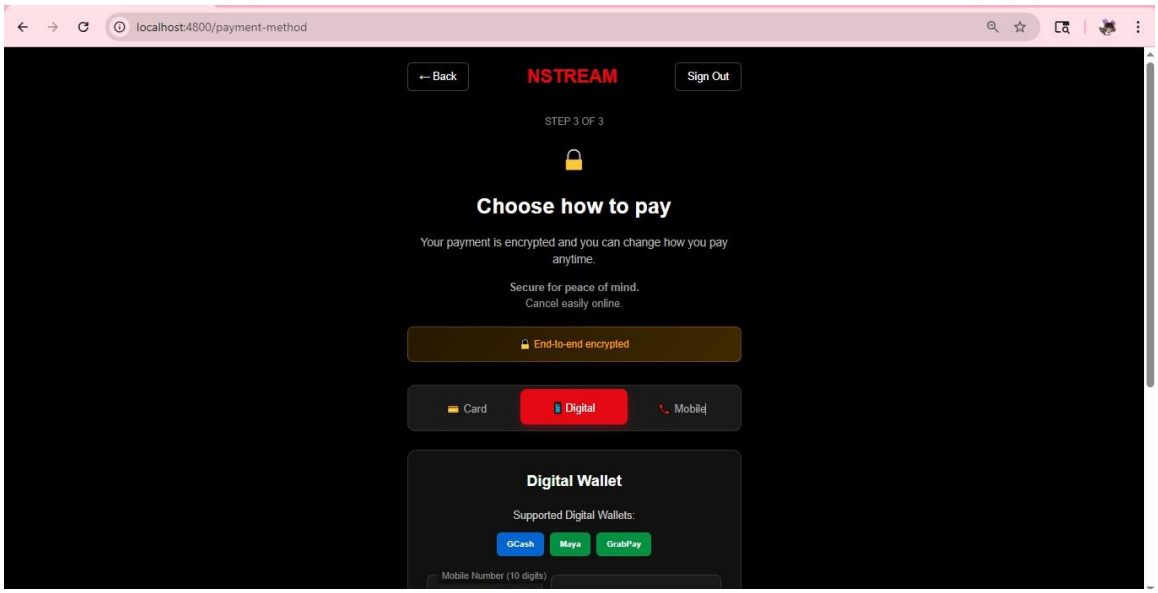
   Open the NStream website via browser and enter the following credentials to sign up. If login is required, enter your registered email and password.

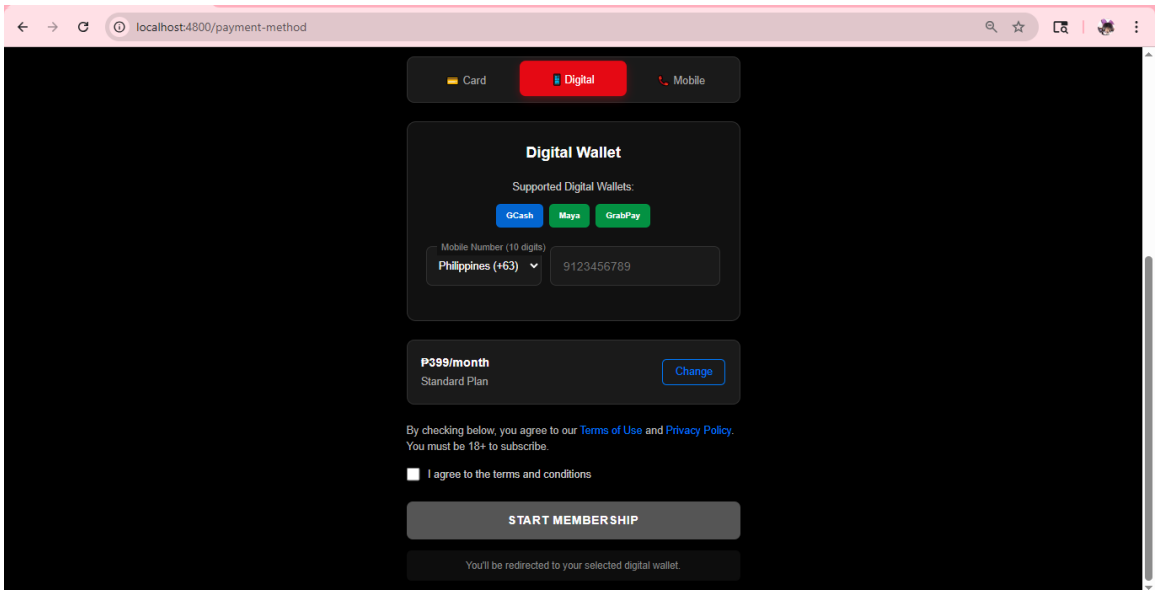

Netnam
Sign In
Unlimited movies, TV shows, and more
Watch anywhere. Cancel anytime.
Ready to watch? Enter your email to create or restart your membership.
donjoaquin629@gmail.com
Get Started


Netnam
Sign Up
Name
ramon
Email
donjoaquin629@gmail.com
Password
Confirm Password
Sign Up
Already have an account? Sign in.



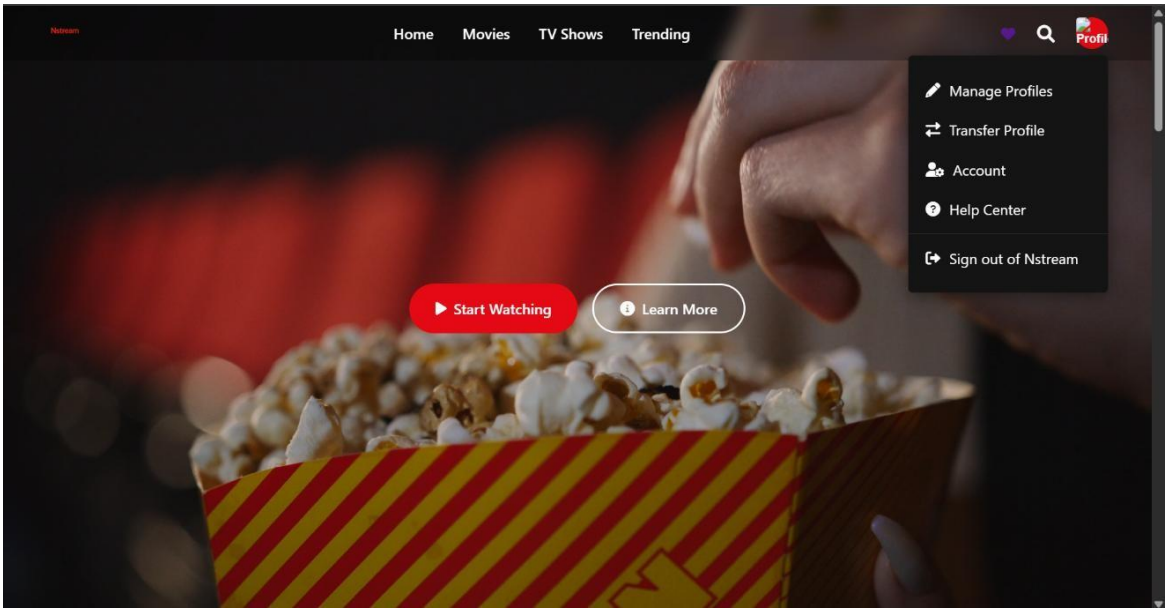
The user will choose the mode of payment of their subscription plans

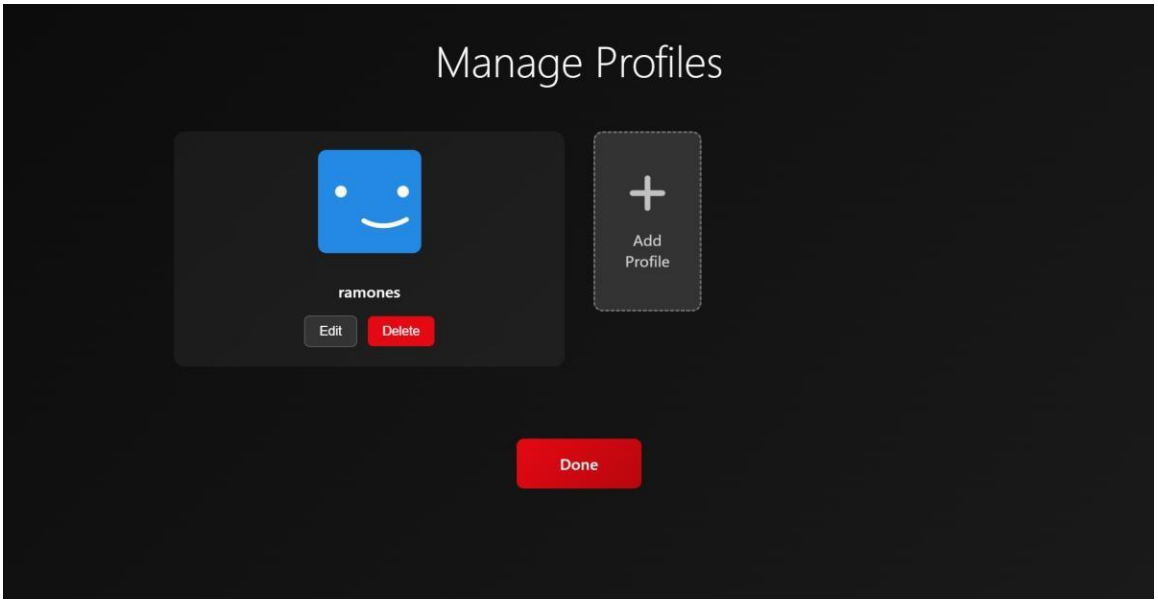

localhost:4800/payment-method
← Back
NSTREAM
Sign Out
STEP 3 OF 3
Choose how to pay
Your payment is encrypted and you can change how you pay anytime.
Secure for peace of mind.
Cancel easily online.
End-to-end encrypted
Card
Digital
Mobile
Digital Wallet
Supported Digital Wallets:
GCash
Maya
GrabPay
Mobile Number (10 digits)


localhost:4800/payment-method
Card
Digital
Mobile
Digital Wallet
Supported Digital Wallets:
GCash
Maya
GrabPay
Mobile Number (10 digits)
Philippines (+63)
9123456789
P399/month
Standard Plan
Change
By checking below, you agree to our Terms of Use and Privacy Policy.
You must be 18+ to subscribe.
I agree to the terms and conditions
START MEMBERSHIP
You'll be redirected to your selected digital wallet.

2. Mainpage

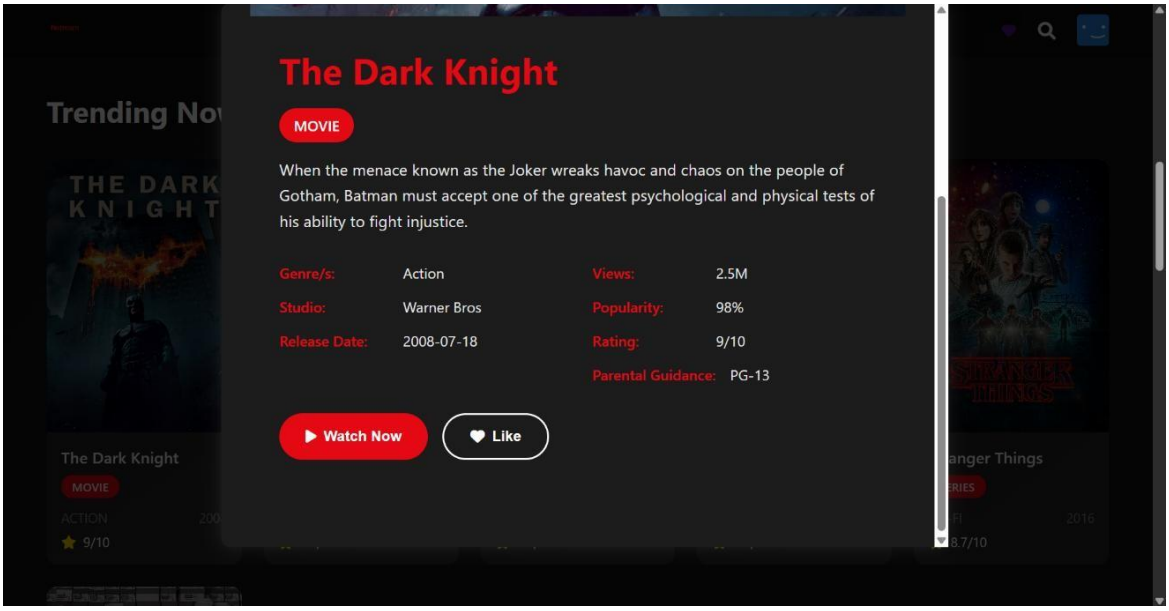
Mainpage will appear. Click “manage profiles” to have a profile and make one or multiple profiles.

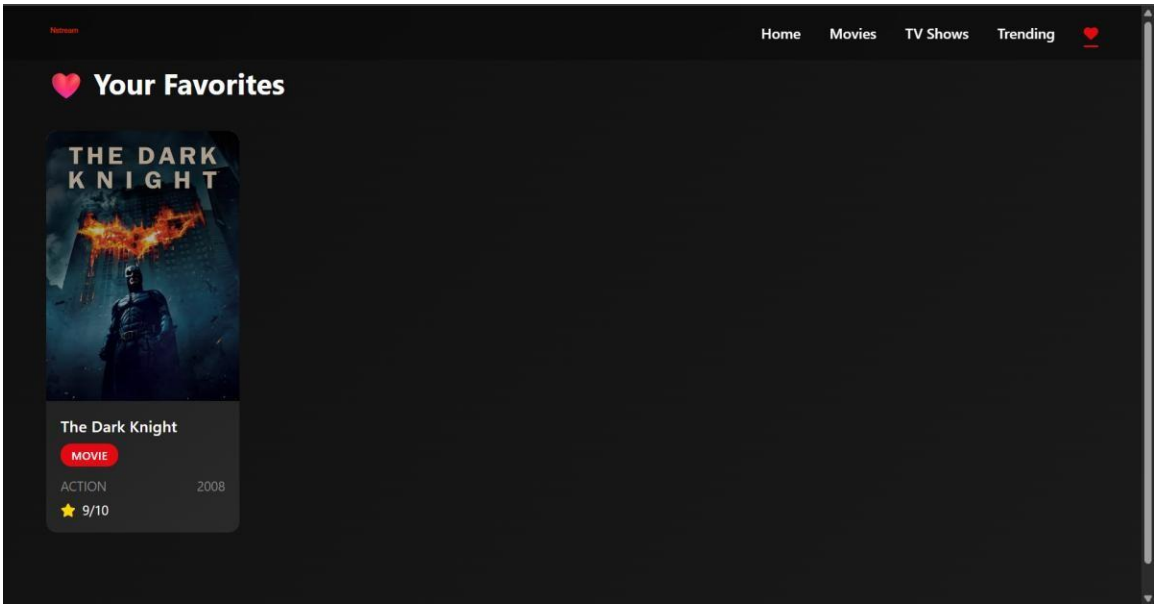




3. Stream

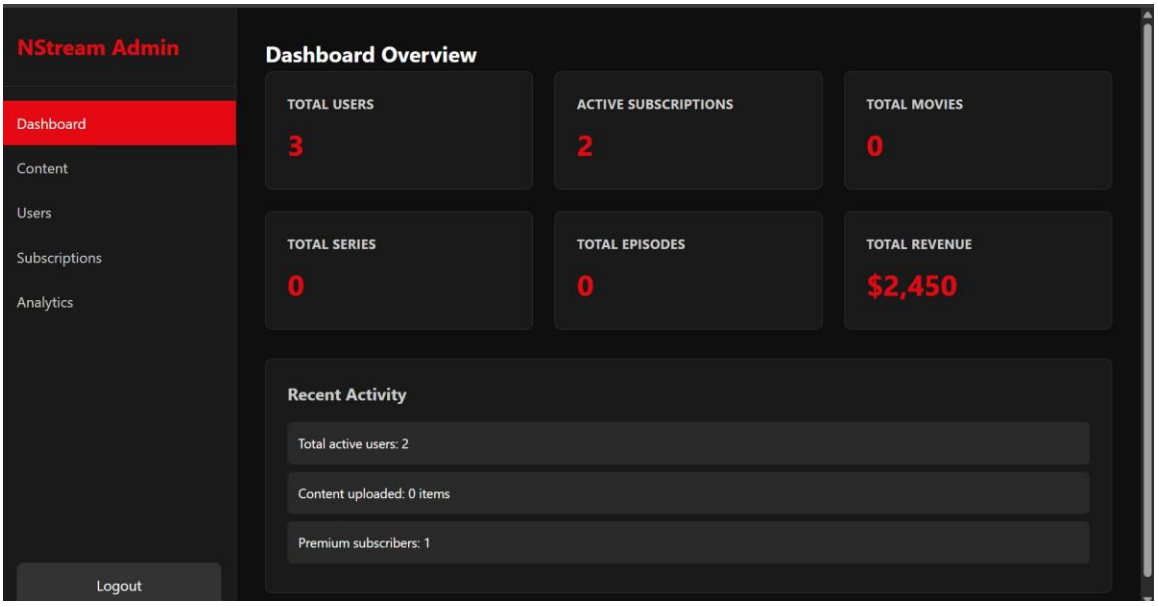
Users can now watch a movie/series or add it to their favorites just clicking the “like” button.

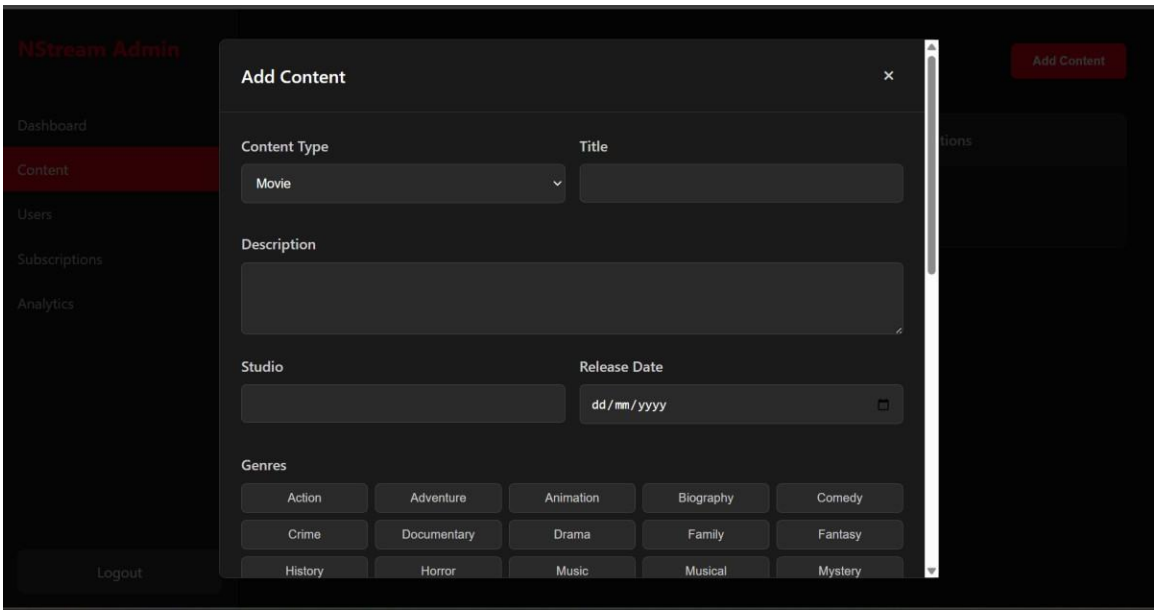


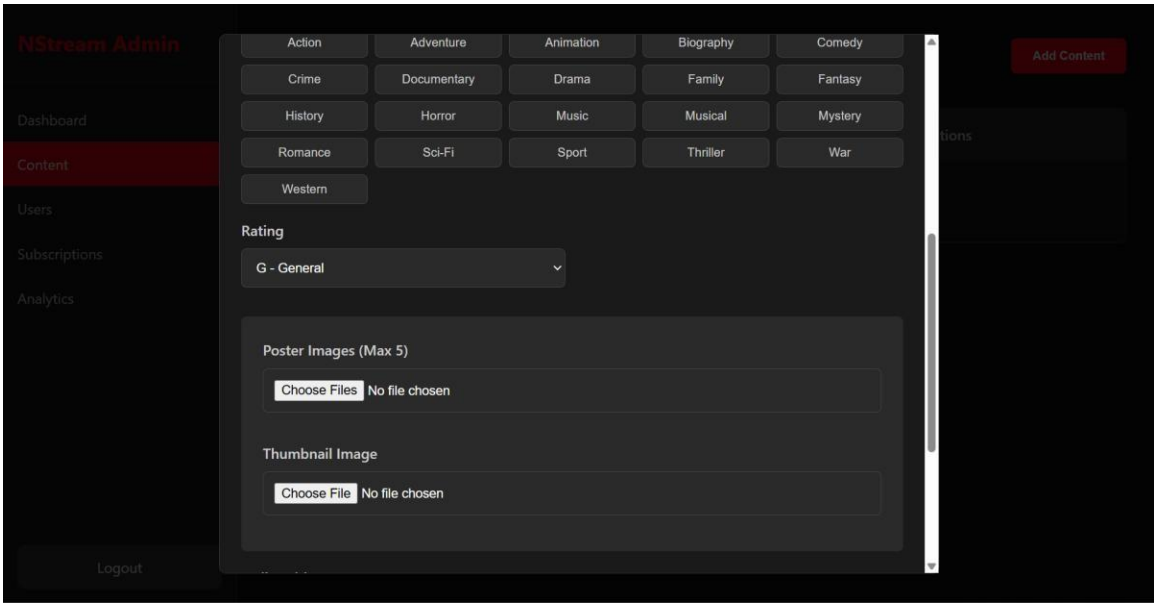


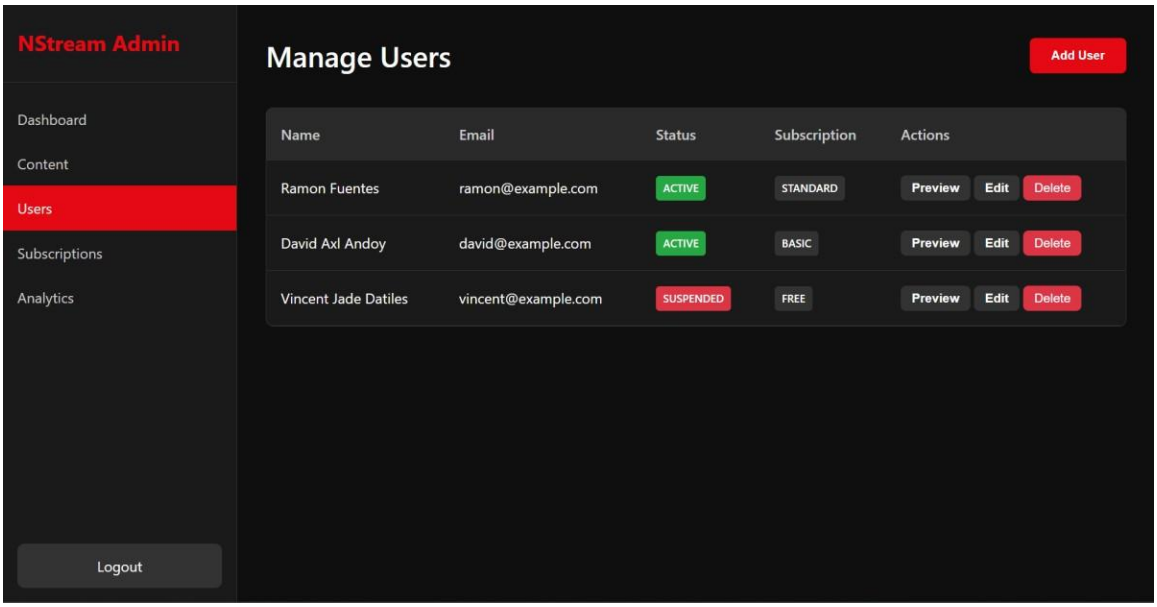

Main Background
Demo • HD • 2024
WARNER BROS. PICTURES
A TimeWarner Company
0:12 / 1:39

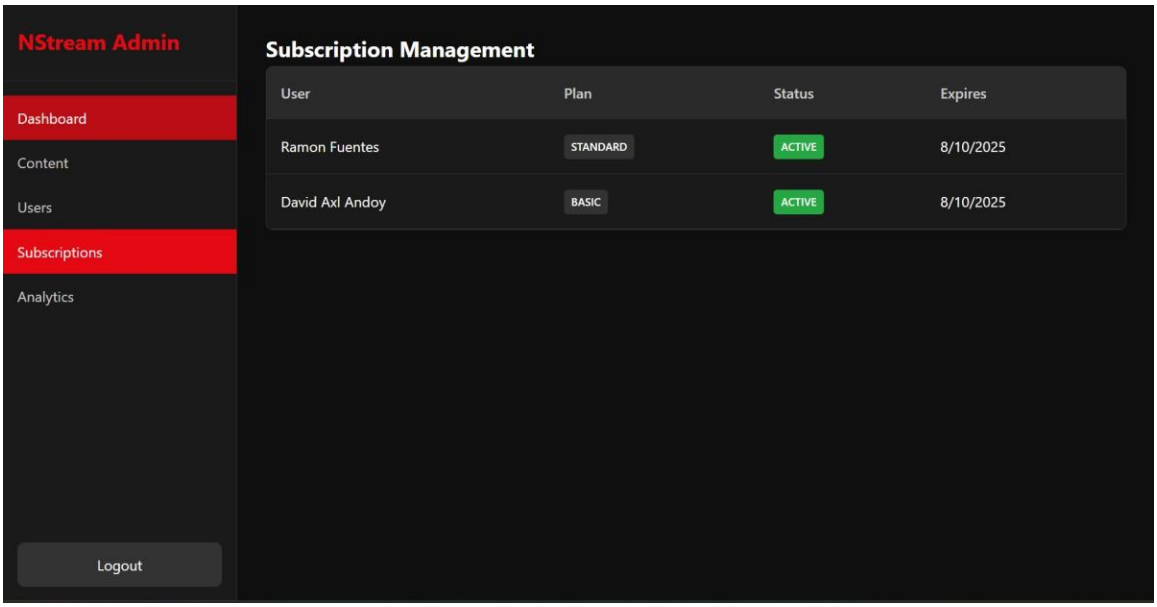
Admin


NStream Admin
Dashboard
Content
Users
Subscriptions
Analytics
Logout
Dashboard Overview
TOTAL USERS
3
ACTIVE SUBSCRIPTIONS
2
TOTAL MOVIES
0
TOTAL SERIES
0
TOTAL EPISODES
0
TOTAL REVENUE
$2,450
Recent Activity
Total active users: 2
Content uploaded: 0 items
Premium subscribers: 1


NStream Admin
Dashboard
Content
Users
Subscriptions
Analytics
Logout
Add Content
Add Content
Content Type
Movie
Title
Description
Studio
Release Date
dd/mm/yyyy
Genres
Action
Adventure
Animation
Biography
Comedy
Crime
Documentary
Drama
Family
Fantasy
History
Horror
Music
Musical
Mystery


NStream Admin
Dashboard
Content
Users
Subscriptions
Analytics
Logout
Add Content
Action
Adventure
Animation
Biography
Comedy
Crime
Documentary
Drama
Family
Fantasy
History
Horror
Music
Musical
Mystery
Romance
Sci-Fi
Sport
Thriller
War
Western
Rating
G - General
Poster Images (Max 5)
Choose Files
No file chosen
Thumbnail Image
Choose File
No file chosen





Conclusion

The development of NStream demonstrates the potential of modern web technologies to deliver a cost-effective, user-friendly, and scalable streaming platform. Designed to resemble the user experience of major platforms like Netflix, NStream prioritizes accessibility and simplicity without compromising on functionality.

By integrating a clean interface, dynamic content rendering, user profile management, and a document-based database using MongoDB, the project provides an ideal solution for students and budget-conscious users seeking ad-free entertainment. Key features such as content filtering, favorites management, and admin-level content control reflect thoughtful system design and user-centric development.

Through this project, the development team successfully applied practical skills in full-stack development, user interface design, and database integration, culminating in a functional platform that addresses real-world needs. NStream serves not only as an academic milestone but also as a foundation for future enhancements—such as real-time streaming, payment integration, or mobile app deployment.

This project showcases both technical capability and creative problem-solving, offering a solid proof-of-concept for lightweight, community-oriented streaming solutions.