i created a "Enhanced RetroCinema Navigator - Your ultimate video experience" . what it does is 1) you can paste any URL into the user box and you will load the video into the VIdeo player. once loaded you can open the FULL CineMatic Experience. https://pdurjsoo.manus.space/ the Cinematic experience mainly was a way to watch any video with no distractions from the outer bezel and frame elements and advertisements. we then added a few extra things like 1) ability to change the size of the viewing screen from really small to full screen and 2) ability to change themes , which change the background around the viewing screen from Home, Movie theater and Drive in. These themes all have elements of each location to bring the viewer the feeling of either "home" or "a movie theater seating and pop corn" or "drive in theater with sound box coming from a extra speaker or radio in car and a bigger screen. these are just extra element added to the area outside the frame of the viewing screen. 3) there is also pre-loaded videos from my youtube channel both playlists a) my prequels https://www.youtube.com/watch?v=slNjuRKN1-U&list=PLd1rdbpKgWxRA8sky8S5H0W0zpSF8rEcB and my b) Foundation videos b) https://www.youtube.com/watch?v=M17P5U9NBZ4&list=PLd1rdbpKgWxTw-pfKMVrFXwpylJur6yjC please look at the web-page and see what other improvements can be made.. i tried to remake this with loveable.ai and it was unable to create without errors or issues. i need to have this working wtihout issues or errors. is there a way to create loading of playlist from youtube or do videos have to be loaded one at a time or to have preloaded videos like we have here do i have to give you every link one at a time like i did in creation of this page the 1st time. please research this page and other abilities we have while keeping the page a static web page and mostly having links from youtube that we can view on this page without external link and sending the viewer to youtube. lets improve this: https://pdurjsoo.manus.space/ . once you have all your research complete , please give me all the complete prompts needed for Manis.ai and bolt.ai and even firebase studio to complete this build. all prompts needed and please go over and above what is needed to make sure the page is innovative in design and appearance is futuristic as if we are in a futuristic video store with the ability to watch videos from any source. all we need is the video URL. lets build this and create all the prompts please here is another example , except this page has issues when loading a URL , it can mostly play preloaded videos from my youtube channel. notice if thumbnail in each video box is clicked, the thumbnail enlarges and we stay on the page. . also if we click to play the video, we also stay on the page and can watch the video or close it and stay on the page. please make sure all of these functions are on the page you built with the prompting. here: https://studio--retrocinema-navigator.us-central1.hosted.app/

Detailed instructions and prompts:

Complete RetroCinema Navigator 2.0 Development Prompts

Gary, here's your comprehensive collection of copy-paste prompts to transform your video player into a futuristic, error-free experience. These prompts are structured to work seamlessly across Manis.ai, Bolt.ai, and Firebase Studio.

How to Use These Prompts

Start with Manis.ai for research and competitive analysis, then move to Bolt.ai for code generation, and finally Firebase Studio for deployment. Each prompt builds on the previous one.

RESEARCH ANALYSIS

```
text

Conduct comprehensive research on the following topics for a
futuristic video player application:

1. YouTube Data API v3 integration:
- Playlist fetching and management
- Video metadata extraction
- Thumbnail generation
- Error handling for restricted content
- Rate limiting and quota management

2. Modern HTML5 video player features:
- Picture-in-picture API implementation
- Video quality selection
```

- Speed control and keyboard shortcuts
- Subtitle/caption support
- Custom video controls

3. Progressive Web App (PWA) capabilities:
- Service worker implementation for video caching
- Offline video playback strategies
- Cache management for large video files
- Background sync for playlist updates

4. Futuristic UI/UX design patterns:
- Glassmorphism and neumorphism design trends
- Neon color schemes and glowing effects
- Particle backgrounds and animations
- Responsive grid layouts for video thumbnails

5. Voice and gesture controls:
- Web Speech API integration
- Voice commands for video playback
- Touch gestures for mobile devices
- Accessibility features for disabled users

Please provide detailed technical specifications, code examples, and best practices for each area.

COMPETITIVE ANALYSIS

text

Analyze the following video player platforms and extract key features for inspiration:

1. YouTube's video player interface
2. Netflix's video streaming interface
3. Vimeo's professional video player

```
4. Twitch's live streaming interface
5. Disney+ streaming experience
6. VLC media player features
7. Modern video conferencing platforms

Focus on:
- User interface design patterns
- Video control mechanisms
- Playlist management systems
- Theme customization options
- Performance optimization techniques
- Accessibility features
- Mobile responsiveness

Provide screenshots, feature comparisons, and implementation
recommendations.
```

BOLT AI PROMPTS

INITIAL SETUP

```
text

Create a modern React application for a futuristic video player
called "RetroCinema Navigator 2.0" with the following
specifications:

TECH STACK:
- React 18 with TypeScript
- Tailwind CSS for styling
- Framer Motion for animations
- React Query for data fetching
- Zustand for state management
- React Router for navigation
```

```
CORE FEATURES:
1. URL input field for any video source
2. YouTube playlist integration using YouTube Data API v3
3. Three immersive themes: Home, Movie Theater, Drive-In
4. Fullscreen cinematic mode with no distractions
5. Video size slider from mini to fullscreen
6. Thumbnail gallery with hover effects
7. Custom video controls with modern design

DESIGN REQUIREMENTS:
- Dark theme with neon accents (#00ffff, #ff00ff, #ffff00)
- Glassmorphism effects for panels
- Particle background animations
- Smooth transitions and micro-interactions
- Mobile-first responsive design
- Accessibility compliance (WCAG 2.1)

UI COMPONENTS NEEDED:
- VideoPlayer component with custom controls
- ThemeSelector component
- PlaylistManager component
- ThumbnailGrid component
- URLInput component
- SizeSlider component
- NavigationHeader component

Start with the basic project structure and implement the video
player component first.
```

ADVANCED FEATURES

```
text
```

```
Enhance the RetroCinema Navigator with advanced features:
```

1. YOUTUBE PLAYLIST INTEGRATION:
- Implement YouTube Data API v3 client
- Fetch playlist items with pagination
- Cache playlist data locally
- Handle API rate limits and errors
- Auto-play next video in playlist
- Shuffle and repeat modes

2. PICTURE-IN-PICTURE MODE:
- Implement browser PiP API
- Custom PiP controls
- Maintain playback state
- Handle window resize events

3. VOICE CONTROLS:
- Web Speech API integration
- Voice commands: "play", "pause", "next", "previous", "volume up/down"
- Visual feedback for voice recognition
- Fallback for unsupported browsers

4. ADVANCED VIDEO FEATURES:
- Video quality selection
- Playback speed control (0.5x to 2x)
- Keyboard shortcuts (space, arrow keys, etc.)
- Video scrubbing and seeking
- Volume control with mute
- Fullscreen toggle

5. THEME CUSTOMIZATION:
- Home theme: Cozy living room with ambient lighting
- Theater theme: Cinema seats with popcorn animations
- Drive-in theme: Car dashboard with radio effects
- Dynamic background particles

- Theme-specific sound effects

6. PERFORMANCE OPTIMIZATIONS:
- Lazy loading for video thumbnails
- Virtual scrolling for large playlists
- Video preloading strategies
- Error boundaries and fallbacks
- Loading states and skeletons

Implement these features incrementally with proper error handling and user feedback.

PERFORMANCE POLISH

text

Optimize and polish the RetroCinema Navigator application:

1. PERFORMANCE OPTIMIZATIONS:
- Implement React.memo for components
- Use useMemo and useCallback hooks
- Optimize bundle size with code splitting
- Implement virtual scrolling for video lists
- Add service worker for caching
- Optimize images and videos for web

2. PWA FEATURES:
- Add web app manifest
- Implement service worker for offline functionality
- Add install prompt
- Cache critical resources
- Background sync for playlists

3. ACCESSIBILITY IMPROVEMENTS:
- Add ARIA labels and roles

- Implement keyboard navigation
- Add focus indicators
- Support screen readers
- Add high contrast mode
- Implement reduced motion preferences

4. ERROR HANDLING:
- Comprehensive error boundaries
- Network error handling
- Video loading error states
- API error handling with retry logic
- User-friendly error messages

5. TESTING:
- Unit tests for components
- Integration tests for API calls
- E2E tests for user flows
- Performance testing
- Accessibility testing

6. DEPLOYMENT PREPARATION:
- Environment configuration
- Build optimization
- Security headers
- Analytics integration
- Error monitoring setup

Ensure the application is production-ready with comprehensive testing and monitoring.

FIREBASE STUDIO PROMPTS

FULL STACK ARCHITECTURE

text

Design and implement a full-stack architecture for RetroCinema Navigator using Firebase services:

FRONTEND ARCHITECTURE:
- React 18 with TypeScript
- Tailwind CSS for styling
- Framer Motion for animations
- React Query for API state management
- React Router for client-side routing

BACKEND SERVICES:
1. Firebase Authentication:
- User registration and login
- Social authentication (Google, Facebook)
- Anonymous authentication for guests
- User profile management

2. Cloud Firestore:
- User preferences and settings
- Custom playlists and favorites
- Watch history and progress tracking
- User-generated content (reviews, ratings)

3. Firebase Storage:
- User avatar uploads
- Custom thumbnail uploads
- Video file uploads (if needed)
- Cached video metadata

4. Cloud Functions:
- YouTube API proxy to hide API keys
- Playlist synchronization
- User analytics processing
- Content moderation

5. Firebase Hosting:
- Static asset hosting
- CDN distribution
- SSL certificate management
- Custom domain setup

DATABASE SCHEMA:

users: {
uid: string,
displayName: string,
email: string,
preferences: {
theme: 'home' | 'theater' | 'drive-in',
autoplay: boolean,
volume: number,
quality: string
},
playlists: PlaylistReference[],
watchHistory: VideoReference[],
createdAt: timestamp,
updatedAt: timestamp
}
playlists: {
id: string,
name: string,
description: string,
videos: VideoReference[],
isPublic: boolean,
createdBy: string,
createdAt: timestamp,
updatedAt: timestamp
}
videos: {
id: string,
title: string,

description: string,
thumbnail: string,
duration: number,
url: string,
source: 'youtube' | 'custom',
metadata: object,
addedAt: timestamp
}
text

SECURITY RULES:
- User data privacy protection
- Playlist access control
- Video content validation
- API rate limiting

Implement this architecture with proper error handling and scalability considerations.

PRODUCTION DEPLOYMENT

text

Deploy RetroCinema Navigator to production using Firebase services:

1. FIREBASE PROJECT SETUP:
- Create Firebase project
- Enable required services (Auth, Firestore, Storage, Functions, Hosting)
- Configure API keys and security rules
- Set up environment variables

2. BUILD OPTIMIZATION:
- Optimize React build for production

- Implement code splitting and lazy loading
- Minimize bundle size
- Configure service worker for caching
- Add compression and minification

3. HOSTING CONFIGURATION:
- Configure Firebase Hosting
- Set up custom domain
- Configure SSL certificates
- Implement CDN caching strategies
- Set up redirects and rewrites

4. CLOUD FUNCTIONS DEPLOYMENT:
- Deploy YouTube API proxy functions
- Set up scheduled functions for maintenance
- Configure function triggers
- Implement proper error handling
- Set up monitoring and logging

5. SECURITY CONFIGURATION:
- Configure Firestore security rules
- Set up authentication rules
- Implement CORS policies
- Configure API key restrictions
- Set up DDoS protection

6. MONITORING AND ANALYTICS:
- Set up Firebase Analytics
- Configure Performance Monitoring
- Implement error tracking
- Set up real-time monitoring
- Configure alerts and notifications

7. CI/CD PIPELINE:
- Set up GitHub Actions workflow

```
- Implement automated testing
- Configure staging environment
- Set up deployment automation
- Implement rollback strategies

8. PERFORMANCE OPTIMIZATION:
- Configure caching strategies
- Optimize database queries
- Implement lazy loading
- Set up content delivery network
- Optimize images and assets

Ensure the deployment is secure, scalable, and monitored for
production use.
```

SPECIALIZED FEATURE PROMPTS

YOUTUBE API INTEGRATION

```
text

YOUTUBE DATA API V3 INTEGRATION PROMPT:

Create a comprehensive YouTube API integration module with the
following features:

1. API CLIENT SETUP:
- Initialize YouTube Data API v3 client
- Handle API key authentication
- Implement rate limiting and quota management
- Error handling for API failures

2. PLAYLIST OPERATIONS:
- Fetch playlist by ID: `playlists.list`
```

- Get playlist items: `playlistItems.list`
- Handle pagination for large playlists
- Extract video metadata (title, description, thumbnail, duration)
- Cache playlist data locally for offline access

3. VIDEO OPERATIONS:
- Get video details by ID: `videos.list`
- Extract video statistics (views, likes, etc.)
- Handle restricted content and region blocks
- Get video captions if available

4. SEARCH FUNCTIONALITY:
- Search videos by keyword: `search.list`
- Filter by duration, upload date, quality
- Get related videos and suggestions
- Implement search history and suggestions

5. ERROR HANDLING:
- Handle quota exceeded errors
- Manage network failures gracefully
- Implement exponential backoff for retries
- User-friendly error messages

Code structure:

interface YouTubeAPIClient {
fetchPlaylist(playlistId: string): Promise<Playlist>
fetchPlaylistItems(playlistId: string, pageToken?: string): Promise<PlaylistItems>
fetchVideoDetails(videoIds: string[]): Promise<VideoDetails[]>
searchVideos(query: string, options?: SearchOptions): Promise<SearchResults>
}
text

undefined

CUSTOM VIDEO PLAYER

```
text
```

```
CUSTOM HTML5 VIDEO PLAYER PROMPT:

Build a fully-featured custom video player component with modern
controls:

1. PLAYER CORE:
- HTML5 video element wrapper
- Custom control overlay
- Responsive design for all screen sizes
- Keyboard shortcuts (space, arrows, etc.)
- Mouse/touch gesture support

2. PLAYBACK CONTROLS:
- Play/pause button with smooth animations
- Progress bar with scrubbing capability
- Time display (current/total)
- Volume control with mute toggle
- Playback speed selector (0.25x to 2x)
- Fullscreen toggle

3. ADVANCED FEATURES:
- Picture-in-picture mode
- Auto-hide controls on inactivity
- Loading spinner and buffering indicators
- Error state handling
- Subtitle/caption support
- Video quality selection

4. VISUAL DESIGN:
- Glassmorphism control panel
- Neon accent colors for interactive elements
- Smooth hover animations
```

```
- Progress bar with glow effects
- Custom icons using Lucide React

5. ACCESSIBILITY:
- ARIA labels for all controls
- Keyboard navigation support
- Screen reader compatibility
- Focus indicators
- Reduced motion preferences

Component structure:
```

interface VideoPlayerProps {
src: string
poster?: string
autoplay?: boolean
loop?: boolean
muted?: boolean
onTimeUpdate?: (currentTime: number) => void
onEnded?: () => void
className?: string
}

```
text

undefined
```

THEME SYSTEM

```
text

IMMERSIVE THEME SYSTEM PROMPT:

Create three distinct immersive themes for the video player:

1. HOME THEME:
- Background: Cozy living room with warm lighting
```

- Elements: Fireplace animation, bookshelves, comfortable furniture
- Color palette: Warm oranges, browns, soft whites
- Ambient sounds: Crackling fire, distant rain (optional)
- UI styling: Wooden textures, soft shadows

2. MOVIE THEATER THEME:
- Background: Classic cinema interior
- Elements: Red velvet seats, movie screen frame, curtains
- Animations: Floating popcorn, subtle curtain movement
- Color palette: Deep reds, golds, dark grays
- Ambient sounds: Theater ambiance, distant crowd murmur
- UI styling: Vintage movie theater aesthetics

3. DRIVE-IN THEME:
- Background: Night sky with stars, car silhouettes
- Elements: Large outdoor screen, car dashboard view
- Animations: Twinkling stars, subtle car movement
- Color palette: Deep blues, purples, bright screen glow
- Ambient sounds: Night crickets, distant car engines
- UI styling: Retro car dashboard controls

IMPLEMENTATION FEATURES:
- Smooth theme transitions using Framer Motion
- Dynamic particle systems for each theme
- Theme-specific sound effects (optional)
- Persistent theme selection in localStorage
- Mobile-optimized versions of each theme

Theme structure:

interface Theme {
id: 'home' | 'theater' | 'drive-in'
name: string
background: BackgroundConfig

particles: ParticleConfig
colors: ColorPalette
sounds?: SoundConfig
animations: AnimationConfig
}
```
text
```

```
undefined
```

Quick Implementation Strategy

1. Start with Manis.ai Research (30 minutes)
    - Run the research prompts to gather technical specs
    - Save all code snippets and best practices
2. Build with Bolt.ai (2-3 hours)
    - Use Initial Setup prompt first
    - Add Advanced Features incrementally
    - Test each feature as you build
3. Deploy with Firebase Studio (1 hour)
    - Set up full-stack architecture
    - Deploy to production with monitoring
4. Iterate and Polish (ongoing)
    - Use specialized prompts for specific features
    - Test across devices and browsers
    - Gather user feedback and improve

These prompts will give you a production-ready, futuristic video player that eliminates all the issues from your current version. The modular approach lets you build piece by piece without overwhelming any single AI tool.

Or you can try this

Updated Prompts for Firebase Studio

Since Firebase Studio is the actual platform, here are optimized prompts specifically designed for its App Prototyping Agent:

FIREBASE STUDIO APP PROTOTYPING PROMPTS

PRIMARY VIDEO PLAYER PROMPT

```
text

Create a futuristic video streaming platform called "RetroCinema
Navigator 2.0" with these specific requirements:

CORE FUNCTIONALITY:
- Universal video URL input that accepts YouTube, Vimeo, direct
video files, and streaming URLs
- YouTube playlist integration using embedded API calls for
seamless playback
- Custom HTML5 video player with picture-in-picture, speed
controls, and keyboard shortcuts
- Three immersive themes: Home (cozy living room), Theater
(cinema experience), Drive-In (outdoor movie night)
- Video size slider from thumbnail to fullscreen with smooth
transitions
- Thumbnail gallery with hover effects and click-to-play
functionality
- Error-free video loading with fallback mechanisms for
restricted content

DESIGN REQUIREMENTS:
- Dark futuristic theme with neon accents (#00ffff, #ff00ff,
#ffff00)
- Glassmorphism panels with subtle transparency and blur effects
- Particle background animations that respond to user
interaction
- Mobile-first responsive design with touch gestures
```

```
- Accessibility compliance with ARIA labels and keyboard
navigation
- Loading animations and smooth micro-interactions throughout

TECHNICAL SPECIFICATIONS:
- Next.js 14 with TypeScript and App Router
- Tailwind CSS for styling with custom neon utilities
- Framer Motion for animations and transitions
- React Query for video metadata caching
- Local storage for user preferences and watch history
- Progressive Web App features with offline playlist access
- Service worker for video preloading and caching

USER EXPERIENCE FEATURES:
- Voice commands for play/pause/next/previous using Web Speech
API
- Gesture controls on mobile devices
- Auto-hide controls during playback
- Video quality selection based on connection speed
- Subtitle support with custom styling
- Full keyboard shortcuts (space, arrows, number keys)
- Social sharing capabilities for playlists

Build this as a production-ready web application that feels like
stepping into a sci-fi video store of the future.
```

ENHANCEMENT PROMPTS (Use After Initial Build)

PLAYLIST INTEGRATION ENHANCEMENT

```
text

Enhance the video player with advanced YouTube playlist
functionality:
```

```
Add a playlist manager that can:
- Import entire YouTube playlists by URL or ID
- Display playlist metadata (title, description, video count, duration)
- Show thumbnail grid of all playlist videos
- Enable playlist shuffle and repeat modes
- Auto-advance to next video with countdown timer
- Remember playback position for each video
- Cache playlist data for offline browsing
- Handle private/restricted videos gracefully

Create a playlist sidebar that slides in from the right with:
- Current playlist queue
- Drag-and-drop reordering
- Add/remove videos from queue
- Search within playlist
- Filter by duration or upload date
- Export playlist as shareable link

Implement playlist analytics showing:
- Most watched videos
- Viewing completion rates
- User engagement metrics
- Recommended videos based on history
```

THEME ENHANCEMENT PROMPT

```
text

Expand the theme system with immersive environmental details:

HOME THEME upgrades:
- Animated fireplace with crackling sound effects
- Moving shadows from window light
- Bookshelf with readable titles that change
```

```
- Coffee steam animation
- Seasonal decorations that change monthly
- Ambient lighting that adjusts with video content

THEATER THEME upgrades:
- Animated popcorn that floats during exciting scenes
- Vintage movie posters that rotate
- Red velvet curtains that open/close between videos
- Theater seat silhouettes with subtle movement
- Classic movie projector beam effect
- Intermission countdown with theater sounds

DRIVE-IN THEME upgrades:
- Twinkling stars that move across the sky
- Car headlights that occasionally sweep across
- Radio static between video transitions
- Weather effects (light rain, fog) for atmosphere
- Car dashboard controls as video player interface
- Distant city skyline with blinking lights

Add theme-specific UI elements:
- Custom video controls styled for each theme
- Theme-appropriate loading animations
- Contextual sound effects and ambient audio
- Dynamic color schemes that adapt to video content
```

ADVANCED FEATURES PROMPT

```
text

Add cutting-edge features to make this the ultimate video
experience:

AI-POWERED FEATURES:
- Content-aware thumbnail generation for custom videos
```

- Automatic chapter detection and navigation
- Smart video recommendations based on viewing patterns
- Real-time content analysis for mood-based theming
- Voice-to-text search within video transcripts
- Automated video quality optimization

SOCIAL FEATURES:
- Watch parties with synchronized playback
- Comment system with timestamp linking
- User-generated playlists with ratings
- Social sharing with custom preview cards
- Follow system for playlist creators
- Video bookmarking with personal notes

PERFORMANCE OPTIMIZATIONS:
- Intelligent video preloading based on viewing patterns
- CDN integration for faster video delivery
- Progressive image loading for thumbnails
- Background video processing for smooth transitions
- Memory management for long viewing sessions
- Bandwidth-adaptive streaming quality

ACCESSIBILITY ENHANCEMENTS:
- Screen reader optimization for all interface elements
- High contrast mode for vision impaired users
- Motion reduction preferences
- Keyboard-only navigation mode
- Voice control for hands-free operation
- Customizable UI scaling for different needs

Build these features incrementally with proper error handling and user feedback.

DEPLOYMENT AND MONITORING PROMPT

```
text
```

Prepare the RetroCinema Navigator for production deployment with comprehensive monitoring:

DEPLOYMENT SETUP:
- Configure Firebase App Hosting for global CDN delivery
- Set up custom domain with SSL certificates
- Implement environment-specific configurations
- Configure build optimization for production
- Set up automated testing pipeline
- Enable compression and minification

MONITORING AND ANALYTICS:
- Firebase Analytics for user behavior tracking
- Performance monitoring for video loading times
- Error tracking with detailed stack traces
- User engagement metrics and heatmaps
- Video completion rate analytics
- Bandwidth usage monitoring

SECURITY MEASURES:
- Firebase App Check with reCAPTCHA Enterprise
- Content Security Policy headers
- Rate limiting for API endpoints
- Input sanitization for URL validation
- Secure video token management
- User data privacy compliance

MAINTENANCE FEATURES:
- Automated backup of user preferences
- Database migration scripts for updates
- A/B testing framework for new features
- Feature flags for gradual rollouts
- Health check endpoints for monitoring
- Automated dependency updates

```
Create a production-ready deployment with 99.9% uptime and
lightning-fast global performance.
```

Firebase Studio Usage Strategy

Step-by-Step Implementation

1. Start with Primary Prompt - Paste the main video player prompt into Firebase Studio's App Prototyping agent
2. Test Core Functionality - Ensure video loading and basic themes work perfectly
3. Add Enhancements Incrementally - Use enhancement prompts one at a time
4. Switch to Code View - Fine-tune specific features using Firebase Studio's VS Code interface
5. Deploy with One Click - Use Firebase Studio's built-in deployment to App Hosting

Firebase Studio Advantages Over Other Platforms

Feature	**Firebase Studio**	**Bolt.ai**	**Lovable.ai**
AI Model	Gemini 2.5 Pro [5]	GPT-4	Custom
Deployment	One-click to Firebase [4]	Manual	Manual
Backend Integration	Automatic Firebase services [4]	Third-party	Limited
Code Access	Full VS Code IDE [2]	Limited	Web-based

Cost	Free during preview [6]	Paid tiers	Paid tiers

Key Firebase Studio Capabilities

Firebase Studio's App Prototyping agent can handle complex video applications because it [4]:

- Automatically provisions backend services when your prompt mentions databases or authentication
- Generates production-ready Next.js code with proper TypeScript structure
- Includes AI features out-of-the-box with Gemini API integration
- Supports multimodal prompts including images and drawings for UI mockups
- Provides real-time preview with instant iteration capabilities

The platform is specifically designed for full-stack AI applications [1], making it perfect for your advanced video player requirements.

ADDITIONAL LINKS:

now add these links to the sidebar and footer: 1) Future Ready Link Hub: https://ready-future-hub-life.lovable.app/ 2) The Dr. Recommends Page: https://dr-gee-advice-hub.lovable.app/ 3) Inside/Out Sweatshirt at ETSY.COM : https://www.etsy.com/shop/FutureReadyShop 4) Future Ready Discoveries : www.futurereadydiscoveries.com 5) Own Your Day : www.futurereadyownyourday.com 6) CineMatic - A non-distracting way to watch content link: https://studio--cinematic-ndr9p.us-central1.hosted.app/

lets fix this page 1) fix sidebar toggle on/off button. make sure when pressed the sidebar moves to the left off the page or back on the page if pressed again. please label it and create on both on the sidebar and the main page