**Optimized Cursor Browser UI Prompts for Spotify-Echo Integration**

**Repository Analysis: EchoTune AI Music Discovery Platform**

Based on analysis of the Spotify-echo repository, this is a sophisticated **AI-powered music discovery platform** with:

**Technical Stack**

* **Backend**: Node.js, Express.js, [Socket.io](http://Socket.io)
* **Frontend**: React, Material-UI
* **Database**: MongoDB (primary), SQLite (fallback)
* **AI Integration**: Multi-provider LLM support (OpenAI GPT-4o, Google Gemini 2.0, Claude 3.5)
* **Music APIs**: Spotify Web API with OAuth integration
* **Infrastructure**: Docker, nginx, automated deployment

**Key Features**

* Conversational AI for music discovery
* Advanced recommendation algorithms (collaborative + content-based filtering)
* Real-time analytics dashboard
* Progressive Web App capabilities
* Comprehensive settings management

**Cursor Prompt Templates for GitHub Coding Agent Integration**

**1. Repository Context & MCP Setup Prompt**

You are working on EchoTune AI, an advanced music discovery platform built with Node.js, React, and multiple AI providers (OpenAI, Gemini, Claude). The codebase includes:  
  
- Multi-provider LLM integration in `/src/chat/llm-providers/`  
- Spotify API services in `/src/spotify/`   
- ML recommendation engine in `/src/ml/`  
- React components in `/src/frontend/components/`  
- MongoDB analytics in `/src/api/routes/`  
  
CONTEXT RULES:  
- Always consider the existing multi-LLM architecture when suggesting integrations  
- Maintain compatibility with current Spotify OAuth flow  
- Preserve the analytics dashboard functionality  
- Follow the established error handling patterns in `chatbot.js`  
  
MCP SERVERS TO LEVERAGE:  
- GitHub MCP Server: For repository operations and PR automation  
- Perplexity MCP Server: For music research and trend analysis   
- Memory MCP Server: For conversation context across sessions  
- Fetch MCP Server: For external music API integrations  
  
When making changes, always:  
1. Run existing tests first  
2. Check compatibility with current LLM providers  
3. Maintain the analytics tracking system  
4. Update relevant documentation

**2. Perplexity Research Integration Prompt**

SYSTEM: You are enhancing EchoTune AI with Perplexity API integration for real-time music research.  
  
TASK: Integrate Perplexity Sonar for music discovery research within the existing chat system.  
  
REQUIREMENTS:  
- Add Perplexity provider to `/src/chat/llm-providers/perplexity-provider.js`  
- Follow the existing provider interface pattern from `openai-provider.js`  
- Use Perplexity for queries about:  
 \* Latest music trends and releases  
 \* Artist information and discography   
 \* Genre analysis and recommendations  
 \* Music industry news and insights  
  
INTEGRATION POINTS:  
- Modify `/src/chat/model-registry.js` to include Perplexity Sonar  
- Update `/src/frontend/components/EnhancedChatInterface.jsx` for research mode  
- Add research capabilities to conversation context  
- Maintain existing analytics tracking  
  
CODE STYLE:  
- Follow existing async/await patterns  
- Use the established error handling from `chatbot.js`  
- Maintain compatibility with current conversation flow  
- Add appropriate logging for analytics dashboard  
  
PERPLEXITY USAGE EXAMPLES:  
- "What are the latest indie rock trends in 2025?"  
- "Research artists similar to [current playing track]"  
- "Find upcoming releases in [user's preferred genres]"

**3. GitHub Actions Automation Prompt**

SYSTEM: You are setting up GitHub Actions workflows for EchoTune AI to automate music research and codebase updates.  
  
CURRENT SETUP:  
- Repository: dzp5103/Spotify-echo  
- Deployment: DigitalOcean with Docker  
- Testing: npm test, linting enabled  
- Environment: Node.js, MongoDB, Redis  
  
CREATE WORKFLOWS FOR:  
  
1. \*\*Music Research Automation\*\* (`.github/workflows/music-research.yml`)  
 - Trigger: Weekly schedule + manual dispatch  
 - Use Perplexity API to research music trends  
 - Update recommendation algorithms with new data  
 - Generate weekly music industry insights report  
  
2. \*\*Code Quality with AI Review\*\* (`.github/workflows/ai-code-review.yml`)   
 - Trigger: Pull requests to main  
 - Use Grok-4 via OpenRouter for code analysis  
 - Focus on music recommendation algorithm improvements  
 - Check integration points with Spotify API  
  
3. \*\*Dependency Updates\*\* (`.github/workflows/dependency-research.yml`)  
 - Research new music/AI libraries using Perplexity  
 - Check compatibility with current multi-LLM setup  
 - Auto-create PRs for relevant updates  
  
SECURITY:  
- Store API keys in GitHub Secrets: PPLX\_API\_KEY, OPENROUTER\_KEY  
- Use minimal permissions for tokens  
- Include rate limiting for API calls  
  
INTEGRATION WITH EXISTING:  
- Respect current Docker deployment setup  
- Maintain DigitalOcean deployment workflow   
- Preserve existing test suite and linting

**4. MCP Server Integration Prompt**

You are integrating MCP servers into EchoTune AI for enhanced GitHub coding agent workflows.  
  
CURRENT ARCHITECTURE:  
- Multi-provider chat system with OpenAI, Gemini, Claude  
- Spotify OAuth integration and audio analysis  
- Real-time analytics with MongoDB  
- React frontend with Material-UI  
  
MCP INTEGRATION STRATEGY:  
  
1. \*\*GitHub MCP Server\*\*  
 - Auto-generate issues for music feature requests  
 - Create PRs for recommendation algorithm updates  
 - Manage release notes with music discovery improvements  
  
2. \*\*Perplexity MCP Server\*\*  
 - Research music trends for recommendation tuning  
 - Analyze user feedback patterns  
 - Generate insights for analytics dashboard  
  
3. \*\*Memory MCP Server\*\*   
 - Maintain conversation context across music discovery sessions  
 - Store user music preferences persistently  
 - Track recommendation effectiveness over time  
  
4. \*\*Filesystem MCP Server\*\*  
 - Manage music analysis data files  
 - Handle playlist exports and imports  
 - Organize training data for ML algorithms  
  
CURSOR CONFIGURATION:  
```json  
{  
 "github-music": {  
 "command": "node",  
 "args": ["./mcp-servers/github-mcp.js"],  
 "env": {  
 "GITHUB\_TOKEN": "ghp\_your\_token",  
 "REPO": "dzp5103/Spotify-echo"  
 }  
 },  
 "perplexity-research": {  
 "command": "node",   
 "args": ["./mcp-servers/perplexity-mcp.js"],  
 "env": {  
 "PPLX\_API\_KEY": "pplx\_your\_key"  
 }  
 }  
}

IMPLEMENTATION:

* Create mcp-servers/ directory in project root
* Follow existing error handling patterns
* Integrate with current analytics system
* Add MCP tools to chat interface options

### 5. Music Discovery Validation Prompt

SYSTEM: You are validating and improving EchoTune AI's music discovery algorithms using AI-powered analysis.

VALIDATION FRAMEWORK:  
Current recommendation engine combines:

* Collaborative filtering (user behavior)
* Content-based filtering (audio features)
* ML models in /src/ml/recommendation-engine.js

VALIDATION TASKS:

1. **Algorithm Performance Analysis**
   * Use Perplexity to research latest recommendation system improvements
   * Analyze user engagement metrics from analytics dashboard
   * Compare current algorithms with industry standards
2. **A/B Testing Integration**
   * Set up experiments for different recommendation strategies
   * Use GitHub Actions to automatically test algorithm variants
   * Track success metrics: saves, plays, user ratings
3. **Real-time Quality Monitoring**
   * Monitor recommendation relevance using chat feedback
   * Use MCP servers to flag unusual patterns
   * Auto-adjust parameters based on user satisfaction

CODE QUALITY CHECKS:

* Validate audio feature analysis accuracy
* Test Spotify API error handling robustness
* Ensure recommendation explanations are meaningful
* Monitor response times for music discovery

RESEARCH INTEGRATION:

* Use Perplexity for competitive analysis of music platforms
* Research emerging audio analysis techniques
* Stay updated on Spotify API changes and new features

AUTOMATION:

* Auto-generate reports on recommendation effectiveness
* Create GitHub issues for algorithm improvements
* Schedule regular model retraining based on new data

### 6. Roadmap Management Prompt

You are managing the EchoTune AI roadmap using AI-powered research and GitHub integration.

CURRENT ROADMAP STATUS:  
✅ Phase 1: Enhanced Intelligence (Multi-LLM, Real-time Analytics, PWA)  
🚧 Phase 2: Social Features (Friend recommendations, Collaborative playlists)  
📋 Phase 3: Platform Expansion (Multi-platform, Enterprise features)  
💡 Phase 4: Innovation (Advanced AI, VR/AR, Music therapy)

ROADMAP AUTOMATION:

1. **Research-Driven Planning**
   * Use Perplexity to research music industry trends
   * Identify emerging technologies for integration
   * Analyze competitor features and market gaps
2. **GitHub Project Management**
   * Auto-create roadmap issues with AI-generated descriptions
   * Link features to code changes using GitHub MCP
   * Track progress with automated milestone updates
3. **Stakeholder Updates**
   * Generate weekly progress reports using analytics data
   * Create technical documentation from code changes
   * Maintain changelog with AI-assisted descriptions

INTEGRATION PRIORITIES:

* Multi-platform music service support (Apple Music, YouTube Music)
* Advanced social features with privacy controls
* Enterprise-grade analytics and insights
* Therapeutic music applications research

PROMPT TEMPLATES FOR ROADMAP:

* "Research latest developments in [music tech area]"
* "Analyze feasibility of [proposed feature] integration"
* "Generate technical specification for [roadmap item]"
* "Identify dependencies for [upcoming milestone]"

Use GitHub MCP to automatically:

* Create epic issues for major features
* Link research findings to development tasks
* Update project boards based on progress
* Generate release notes from commits

## Usage Instructions  
  
1. \*\*Copy the appropriate prompt\*\* based on your current task  
2. \*\*Paste into Cursor's Composer\*\* (Cmd/Ctrl + I)  
3. \*\*Add specific context\*\* about what you're working on  
4. \*\*Let the AI provide tailored solutions\*\* for your EchoTune AI codebase  
  
These prompts are optimized for the Spotify-Echo repository's architecture and will provide contextually relevant suggestions while maintaining integration with your existing GitHub coding agent and MCP server workflows.