# SECRET HELIX
# MUSIC WEB APP

Design Specification & Developer Workflow

| | |
|---|---|
| Project Name: | Secret Helix Radio |
| Domain: | secret.whisper.college |
| Type: | Private Music Web Application |
| Access: | Friends & Family Only |
| Document Version: | 1.0 |

# TABLE OF CONTENTS

# 1. PROJECT OVERVIEW

Secret Helix is a private music web application designed as a collaborative radio experience for friends and family. The application allows users to connect their existing music streaming accounts (Spotify, Apple Music, SoundCloud, Pandora) and contribute songs to a shared queue represented by an animated DNA helix visualization.

## Key Differentiators:

- **Uninterruptible Playback:** Like a real radio station, the currently playing song cannot be skipped or interrupted
- **DNA Helix Queue:** Songs are visually represented as elements within a rotating 3D DNA helix structure
- **Multi-Platform Integration:** Users can inject songs from multiple streaming services into one unified queue
- **Private & Exclusive:** Accessible only via secret.whisper.college subdomain to invited users
- **Collaborative Experience:** All connected users see the same queue and hear the same music simultaneously

## 2. CORE CONCEPT & VISION

The Secret Helix creates a shared listening experience that mimics the communal nature of traditional radio while incorporating modern streaming flexibility. The DNA helix metaphor represents the intertwining of different users' musical tastes, creating a unique genetic code of sound that evolves with each contributed track.

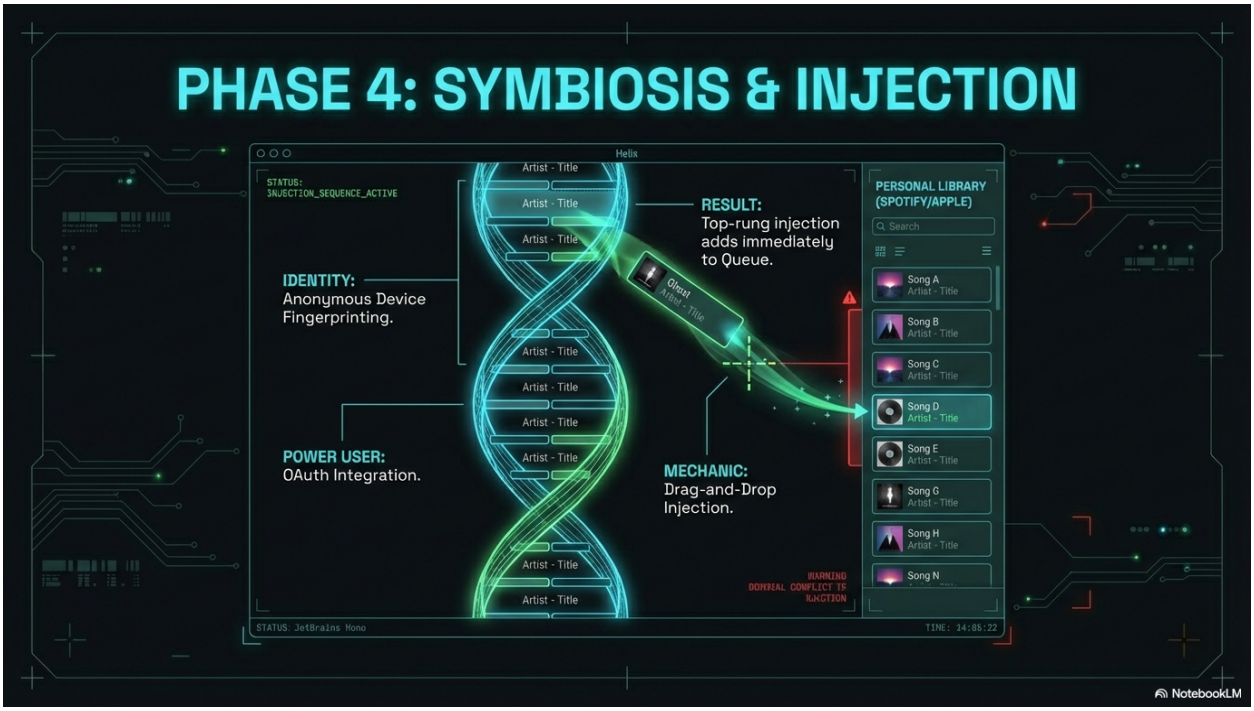
### The Radio Station Principle:

The core mechanic that defines Secret Helix is the uninterruptible nature of the currently playing track. This design choice is intentional and critical to the user experience. Just as traditional radio listeners cannot skip songs, Secret Helix users must respect the song at the "top rung" of the DNA helix. This creates:

- A sense of shared experience and anticipation
- Respect for each contributor's musical choices
- Discovery of songs users might otherwise skip
- A more deliberate and thoughtful queue curation process

# 3. USER INTERFACE DESIGN REFERENCE

The following image represents the closest approximation to the desired user interface aesthetic and functionality. This design should serve as the primary visual reference for development.




## Design Analysis:

- **DNA Helix Visualization:** The central animated helix displays queued tracks on the rungs between the phosphate backbones
- **Dark Cyberpunk Aesthetic:** Black background with cyan/turquoise accent colors creates a futuristic, tech-forward atmosphere
- **Circuit Board Patterns:** Subtle technical diagrams in the background reinforce the digital/technological theme
- **Status Indicators:** Top-left corner shows system status with injection sequence active notification
- **Personal Library Panel:** Right sidebar displays user's available songs from connected streaming services
- **Drag & Drop Mechanic:** Visual indication of songs being injected from personal library into the helix queue
- **Identity & Authentication:** Anonymous device fingerprinting for session management
- **OAuth Integration:** Power user authentication for streaming service connections
- **Result Feedback:** Clear messaging that top-rung injection adds immediately to active queue

# 4. TECHNICAL ARCHITECTURE

## Frontend Stack:

The frontend must support complex 3D animations, real-time updates, and seamless integration with multiple music streaming APIs. The following stack is recommended:

| Component | Technology | Purpose |
|---|---|---|
| Framework | React 18+ | Component-based UI with hooks for state management |
| 3D Rendering | Three.js or Babylon.js | DNA helix animation and 3D scene management |
| Real-time Sync | WebSockets (Socket.io) | Live queue updates across all connected clients |
| State Management | Redux or Zustand | Global state for queue, user sessions, playback |
| Drag & Drop | react-dnd or native HTML5 | Intuitive song injection from library to queue |
| Styling | Tailwind CSS or styled-components | Consistent cyberpunk aesthetic |

## Backend Stack:

The backend manages user authentication, streaming service integration, queue synchronization, and real-time communication between clients.

| Component | Technology | Purpose |
|---|---|---|
| Runtime | Node.js + Express | Fast, event-driven API server |
| Database | PostgreSQL or MongoDB | User accounts, queue persistence, session data |
| Real-time | Socket.io | Bidirectional queue updates and playback sync |
| Authentication | OAuth 2.0 + JWT | Secure integration with streaming services |
| Caching | Redis | Session storage and queue state caching |

## Streaming Service Integration:

Users must be able to connect one or more of the following services. Each requires OAuth 2.0 authentication and has specific API capabilities and limitations:

| Service | API Capabilities | Limitations |
|---|---|---|
| Spotify | Full playback control, search, playlists | Premium account required for playback |
| Apple Music | Search, catalog access, playback | Requires MusicKit JS, Apple Developer account |

| | | |
|---|---|---|
| SoundCloud | Track streaming, user playlists | Limited to publicly available tracks |
| Pandora | Station creation, thumbs up/down | Limited direct playback control |

# 5. FEATURE BREAKDOWN

## 5.1 DNA Helix Queue Visualization

The DNA helix is the centerpiece of the user interface and must be implemented with attention to both aesthetic and functional requirements.

- **3D Animation:** Continuous rotation of the helix structure with smooth 60fps performance
- **Song Placement:** Tracks appear as labeled rungs between the two strands of the helix
- **Queue Order:** Top-most rung is the currently playing track, subsequent rungs are queued in descending order
- **Visual Distinction:** Currently playing track should be visually highlighted (brighter glow, different color)
- **Dynamic Updates:** New songs animate into position when added, smoothly integrating into the helix structure
- **Interaction:** Hovering over a song rung displays additional metadata (artist, album, duration, contributor)
- **Glow Effects:** Cyan/turquoise neon glow effects around the helix strands and active elements
- **Background Integration:** Circuit board patterns fade into the background without competing for visual attention

## 5.2 Personal Library Panel

The personal library sidebar aggregates songs from all connected streaming services into a unified, searchable interface.

- **Multi-Service Display:** Shows songs from Spotify, Apple Music, SoundCloud, Pandora in a single list
- **Search Functionality:** Real-time search across all connected services
- **Album Art Thumbnails:** Small square thumbnails for visual recognition
- **Service Indicators:** Color-coded icons or badges showing which service each song is from
- **Drag-to-Queue:** Users can click and drag songs from library into the DNA helix
- **Playlist Access:** Expandable playlist sections for each connected service
- **Recently Played:** Quick access to recently added songs
- **Scrollable List:** Smooth infinite scroll with lazy loading for large libraries

## 5.3 Drag & Drop Injection Mechanic

The injection mechanic is the primary method users employ to add songs to the queue. It must be intuitive, visually satisfying, and respect the radio station principle.

- **Visual Feedback:** Song card follows cursor during drag, with semi-transparent ghost effect
- **Drop Zones:** Valid drop areas on the helix glow or pulse when hovering with a dragged song

- **Insertion Animation:** Dropped songs animate smoothly into their position in the helix
- **Top-Rung Protection:** Currently playing song position is not a valid drop target
- **Queue Position Options:** Users can drop songs to 'Next Up' position or 'End of Queue'
- **Confirmation Feedback:** Brief notification confirms song added with position in queue
- **Multi-Select:** Advanced feature to select and inject multiple songs at once
- **Conflict Resolution:** If multiple users inject simultaneously, server determines final order

## 5.4 OAuth Integration & User Identity

Users must authenticate with their streaming services to access their libraries. The application also maintains user identity for session management and queue contribution tracking.

- **OAuth 2.0 Flow:** Standard authorization code flow for Spotify, Apple Music, SoundCloud
- **Multi-Service Support:** Users can connect multiple services to the same account
- **Anonymous Fingerprinting:** Device fingerprinting for users who don't want to create accounts
- **Session Persistence:** Refresh tokens stored securely to maintain authentication
- **Permission Scopes:** Request minimal necessary permissions (read library, playback control)
- **Account Linking:** Interface to manage connected services and revoke access
- **Contributor Attribution:** Each queued song displays which user added it
- **Profile Customization:** Users can set display name and avatar for queue attribution

## 5.5 Synchronized Playback

All connected users must hear the exact same audio at the same time, creating a true shared listening experience.

- **Single Source of Truth:** Server maintains authoritative playback state and timestamp
- **WebSocket Broadcasting:** Real-time playback events (play, pause, track change) sent to all clients
- **Time Synchronization:** Clients sync their local playback to server timestamp within 100ms accuracy
- **Buffer Management:** Pre-buffer next track to eliminate gaps between songs
- **Network Tolerance:** Handle brief disconnections gracefully with automatic resync
- **Playback Controls:** Only play/pause available; no skip or seek to enforce radio model
- **Volume Control:** Individual client-side volume without affecting others
- **Now Playing Display:** Prominent display of current track with progress bar

# 6. USER WORKFLOWS

## 6.1 First-Time User Onboarding

1. User navigates to secret.whisper.college
2. Landing page displays Secret Helix branding and brief explanation
3. User chooses to connect streaming service (Spotify, Apple Music, SoundCloud, Pandora)
4. OAuth flow redirects to selected service for authentication
5. User authorizes Secret Helix to access their library
6. Redirect back to app with access token
7. User prompted to set display name and optional avatar
8. Tutorial overlay explains DNA helix, queue mechanics, and drag-to-inject
9. User enters main application interface with personal library loaded

## 6.2 Adding a Song to Queue

1. User searches or browses their personal library in right sidebar
2. User finds desired song and clicks to begin drag operation
3. Song card follows cursor with semi-transparent ghost effect
4. DNA helix drop zones glow to indicate valid injection points
5. User drags song over desired position (Next Up or End of Queue)
6. Drop zone pulses brighter when song is hovering over it
7. User releases mouse to drop song into queue
8. Injection animation shows song integrating into helix structure
9. Brief notification confirms: 'Song added to queue at position 5'
10. Queue updates in real-time for all connected users

## 6.3 Passive Listening Experience

1. User opens Secret Helix in browser
2. Application auto-connects to existing session (or prompts for authentication if needed)
3. DNA helix loads with current queue state
4. Currently playing song is highlighted at top of helix
5. User can see who contributed each queued song via attribution labels
6. Playback continues seamlessly as user browses or minimizes window

7. Visual progress bar shows time remaining on current track
8. When track ends, helix rotates to bring next song to top position
9. New song begins playing automatically without user intervention
10. User can add songs to queue at any time without interrupting current playback

# 7. IMPLEMENTATION PRIORITIES

Development should proceed in phases, with each phase building upon the previous to create a minimum viable product before adding advanced features.

## Phase 1: Core Infrastructure

- **Backend Setup:** Express server, database schema, WebSocket infrastructure
- **Authentication:** OAuth flow for at least one streaming service (recommend Spotify first)
- **Basic Frontend:** React app structure, routing, simple UI without 3D elements
- **Queue Management:** Server-side queue logic, add/remove songs, next track advancement
- **Real-time Sync:** Socket.io connection between clients and server for queue updates

## Phase 2: DNA Helix Visualization

- **3D Engine Integration:** Three.js or Babylon.js setup
- **Helix Geometry:** Create double helix structure with rungs for songs
- **Animation System:** Continuous rotation, smooth interpolation of new song insertion
- **Song Cards:** Display track metadata on helix rungs with proper positioning
- **Visual Effects:** Neon glow, cyberpunk aesthetic, circuit board backgrounds
- **Performance Optimization:** Ensure 60fps on target devices

## Phase 3: Library & Drag-and-Drop

- **Personal Library UI:** Right sidebar with song list from connected service
- **Search Functionality:** Filter songs by title, artist, album
- **Drag-and-Drop:** Implement HTML5 drag API or react-dnd
- **Drop Zones:** Define and visualize valid injection points on helix
- **Injection Animation:** Smooth integration of dropped songs into queue
- **User Feedback:** Tooltips, notifications, visual confirmations

## Phase 4: Synchronized Playback

- **Playback Engine:** Integration with Spotify Web Playback SDK (or equivalent)
- **Server Timestamp:** Authoritative playback state maintained on server
- **Client Synchronization:** Clients sync to server time within acceptable tolerance

- **Auto-Advancement:** Automatic progression to next track when current song ends
- **Playback Controls:** Play/pause only, no skip functionality
- **Now Playing Display:** Current track info, progress bar, contributor attribution

## Phase 5: Multi-Service & Polish

- **Additional OAuth Flows:** Apple Music, SoundCloud, Pandora integrations
- **Service Aggregation:** Combine libraries from multiple services in unified UI
- **Account Management:** Interface for linking/unlinking services
- **Visual Polish:** Animations, transitions, loading states, error handling
- **Responsive Design:** Mobile and tablet layouts (though desktop is primary target)
- **Testing & Debugging:** Cross-browser compatibility, performance profiling

# 8. TECHNICAL REQUIREMENTS & CONSTRAINTS

## Critical Design Rules:

• **Uninterruptible Playback:** The currently playing song CANNOT be skipped, paused by other users, or removed from queue. Only the individual user can pause their own local playback.

• **Queue Integrity:** Songs in queue cannot be reordered once added. Queue follows strict FIFO (first-in-first-out) except for 'Next Up' priority injection.

• **Single Playback Instance:** Only one server-managed playback session exists at a time. All clients are synchronized to this session.

• **Private Access:** Application is accessible ONLY via secret.whisper.college subdomain. No public discovery or indexing.

• **Friends & Family Only:** Invite-only access control. Consider implementing invite codes or whitelist.

## Performance Requirements:

• DNA helix animation must maintain 60fps on modern desktop browsers

• WebSocket latency for queue updates should be under 200ms

• Initial page load time under 3 seconds on broadband connection

• Playback synchronization accuracy within 100ms across clients

• Support for at least 20 simultaneous connected users

• Personal library should load and display 1000+ songs without lag

## Browser Compatibility:

Desktop browsers are the primary target. Mobile support is secondary but should be functional.

| Browser | Minimum Version | Priority |
|---|---|---|
| Chrome | Latest | Primary |
| Firefox | Latest | Primary |
| Safari | Latest | Secondary |
| Edge | Latest | Secondary |
| Mobile Safari | iOS 15+ | Tertiary |
| Mobile Chrome | Android 10+ | Tertiary |

## Security & Privacy:

- **OAuth Token Storage:** Refresh tokens encrypted at rest in database
- **HTTPS Only:** All traffic over TLS, no mixed content
- **CORS Configuration:** Strict origin policy limiting to whisper.college domain
- **Rate Limiting:** Prevent abuse of queue injection and API calls
- **Session Management:** Secure session cookies with httpOnly and sameSite flags
- **API Authentication:** JWT tokens for authenticated backend requests
- **Input Validation:** Sanitize all user inputs to prevent injection attacks

# DEVELOPER NOTES & GUIDANCE

This document represents the product vision for Secret Helix. As the developer, you have creative freedom in implementation details that don't conflict with the core requirements. However, the following elements are non-negotiable and must be implemented as specified:

## Non-Negotiable Requirements:

- DNA helix visualization as the primary queue interface
- Uninterruptible playback of the currently playing track
- Drag-and-drop song injection from personal library to queue
- Real-time synchronization across all connected users
- OAuth integration with multiple streaming services
- Dark cyberpunk aesthetic matching the reference image
- secret.whisper.college domain deployment

## Areas of Creative Freedom:

- Exact implementation of the 3D helix animation (library choice, shader effects)
- Additional UI elements that enhance usability without cluttering the interface
- Specific WebSocket message formats and protocols
- Database schema structure as long as it supports required features
- Error handling and edge case behavior not explicitly defined in this document
- Loading states, transitions, and micro-animations
- Accessibility features and keyboard navigation

## Communication & Iteration:

Development is an iterative process. Please share work-in-progress builds frequently for feedback. When the implementation diverges from the vision described in this document, discuss the reasoning before proceeding. The goal is to maintain alignment while leveraging your technical expertise.

If any aspect of this specification is unclear or seems technically infeasible, please raise questions early rather than making assumptions. The vision is firm but the path to achieving it is collaborative.

This project represents a unique opportunity to build something genuinely novel in the music streaming space. The combination of collaborative queueing, synchronized playback, and the DNA

helix metaphor creates an experience that doesn't exist elsewhere. Thank you for bringing this vision to life.