Based on your requirements and the JSON structure you've shown, here's a comprehensive technical prompt:

**Interactive Podcast Study Tool - Technical Specification**

**Project Overview**

Create a web-based interactive podcast study tool that synchronizes audio playback with word-level transcript highlighting, featuring click-to-seek functionality, playback controls, sentence looping, and a personal dictionary system with color-coded words.

**File Structure & Location**

* Project directory: C:\Users\glaub\OneDrive\Documentos\000\_teste\LinguaSpace
* Audio file: preview\_file.mp3
* Transcript file: Books\_Summary.json
* Development environment: Visual Studio Code

**JSON Data Structure**

{

"segments": [

{

"text": "Full sentence text",

"words": [

{

"text": "word",

"start": 0.079,

"end": 0.219,

"type": "word",

"speaker\_id": "Speaker 0",

"characters": null

}

]

}

]

}

**Core Architecture**

**1. Project Structure**

LinguaSpace/

├── index.html

├── styles.css

├── js/

│ ├── app.js

│ ├── audioPlayer.js

│ ├── transcriptRenderer.js

│ ├── dictionary.js

│ └── utils.js

├── assets/

│ ├── preview\_file.mp3

│ └── Books\_Summary.json

└── data/

└── dictionary.json

**2. HTML Structure**

<!DOCTYPE html>

<html>

<head>

<title>LinguaSpace - Interactive Podcast Study Tool</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div id="app">

<div id="transcript-container">

<div id="transcript-content"></div>

</div>

<div id="controls-panel">

<div id="audio-controls">

<audio id="audio-player"></audio>

<!-- Playback controls here -->

</div>

</div>

</div>

<script type="module" src="js/app.js"></script>

</body>

</html>

**Detailed Implementation Requirements**

**3. Core Features Implementation**

**3.1 Dynamic Word Highlighting**

* Track current playback time using audio.currentTime
* Compare with word timestamps (start and end) from JSON
* Apply light blue background (#ADD8E6) to current word
* Remove highlight when word ends
* Implement smooth auto-scrolling to keep current word visible

**Implementation approach:**

class TranscriptHighlighter {

constructor(audioElement, transcriptData) {

this.audio = audioElement;

this.segments = transcriptData.segments;

this.currentWordIndex = 0;

this.wordElements = [];

}

updateHighlight() {

const currentTime = this.audio.currentTime;

// Find current word based on timestamp

// Apply/remove highlighting

// Auto-scroll if needed

}

}

**3.2 Click-to-Seek**

* Add click event listeners to each word element
* On click, set audio.currentTime to word's start timestamp
* Update highlighting immediately

**3.3 Quick Navigation Controls**

* Create buttons for: -10s, -5s, -3s, -1s, +1s, +3s, +5s, +10s
* Implement seek functionality with bounds checking
* Visual feedback on button press

**3.4 Sentence Loop Feature**

* Define sentence boundaries (between periods)
* Toggle button for "Loop Active"
* When active:
  + Identify current sentence boundaries
  + Set loop start/end points
  + Auto-restart at sentence beginning when reaching end
  + Add 1-second delay between loops

**3.5 Personal Dictionary**

* Right-click context menu on words
* Color picker with 7 predefined colors:
  + Red (#FF6B6B)
  + Orange (#FFA500)
  + Yellow (#FFD700)
  + Green (#4CAF50)
  + Blue (#2196F3)
  + Purple (#9C27B0)
  + Pink (#FF69B4)
* Save to localStorage structure:

{

"dictionary": {

"word": {

"color": "#FF6B6B",

"addedDate": "2024-01-15",

"occurrences": []

}

}

}

**3.6 Playback Speed Control**

* Buttons for: 0.75×, 1×, 1.25×, 1.5×
* Use audio.playbackRate property
* Visual indicator of current speed

**4. UI/UX Specifications**

**4.1 Layout**

* Transcript area: 70% of viewport height
* Control panel: Fixed bottom, 30% height
* Responsive design for different screen sizes

**4.2 Transcript Display**

* Font: Clean, readable (e.g., Inter, Roboto)
* Line height: 1.8 for readability
* Padding between segments
* Smooth scrolling behavior

**4.3 Visual Feedback**

* Hover effects on clickable words
* Button press animations
* Loading states
* Error handling displays

**5. Technical Implementation Details**

**5.1 State Management**

class AppState {

constructor() {

this.currentPodcast = null;

this.isLooping = false;

this.loopBoundaries = { start: 0, end: 0 };

this.playbackSpeed = 1;

this.dictionary = this.loadDictionary();

}

}

**5.2 Event Handling**

* Debounce scroll events
* Throttle time update events (60fps max)
* Proper cleanup on component unmount

**5.3 Performance Optimizations**

* Virtual scrolling for long transcripts
* Efficient DOM updates (batch operations)
* RequestAnimationFrame for smooth animations
* Lazy loading for multiple podcasts

**6. Multi-Podcast Support**

* File picker for loading new podcasts
* Podcast library management
* Shared dictionary across all podcasts
* Progress tracking per podcast

**7. Error Handling**

* Validate JSON structure on load
* Handle missing timestamps gracefully
* Audio loading errors
* LocalStorage quota exceeded

**8. Browser Compatibility**

* Target modern browsers (Chrome, Firefox, Safari, Edge)
* Polyfills for older browser support if needed
* Progressive enhancement approach

**Development Phases**

**Phase 1: Core Functionality**

1. Basic audio player with transcript display
2. Word-level highlighting sync
3. Click-to-seek implementation

**Phase 2: Enhanced Controls**

1. Quick navigation buttons
2. Playback speed control
3. Sentence loop feature

**Phase 3: Dictionary System**

1. Word selection and color coding
2. LocalStorage integration
3. Dictionary management UI

**Phase 4: Polish & Optimization**

1. Smooth animations and transitions
2. Performance optimizations
3. Multi-podcast support
4. Error handling and edge cases

**Testing Requirements**

* Test with various podcast lengths (up to 20 minutes)
* Different timestamp formats
* Browser compatibility testing
* Performance profiling
* User experience testing

**Additional Considerations**

* Keyboard shortcuts for power users
* Export/import dictionary feature
* Dark mode support
* Mobile-responsive design
* Accessibility features (ARIA labels, keyboard navigation)

This tool should provide an immersive language learning experience with precise audio-text synchronization and personalized vocabulary tracking across multiple podcasts.