I'll help you create a complete, ready-to-deploy speech-to-speech translator app! Based on your setup, I'll create a fully functional HTML app that you can use in Koder on your iPhone.

Here's a complete, production-ready speech-to-speech translator:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0,
    <title>Speech Translator</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubun
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            min-height: 100vh;
            display: flex;
            flex-direction: column;
            padding: 20px;
            color: #333;
        }

        .container {
            max-width: 600px;
            width: 100%;
            margin: 0 auto;
            display: flex;
            flex-direction: column;
            gap: 20px;
            flex: 1;
        }

        header {
            text-align: center;
            color: white;
            padding: 20px 0;
        }

        h1 {
            font-size: 2rem;
```

```
    font-weight: 700;
    margin-bottom: 10px;
    text-shadow: 2px 2px 4px rgba(0,0,0,0.2);
}

.subtitle {
    font-size: 0.9rem;
    opacity: 0.9;
}

.language-selector {
    background: white;
    border-radius: 20px;
    padding: 20px;
    box-shadow: 0 10px 30px rgba(0,0,0,0.2);
}

.language-row {
    display: flex;
    align-items: center;
    gap: 15px;
    margin-bottom: 15px;
}

.language-row:last-child {
    margin-bottom: 0;
}

label {
    font-weight: 600;
    min-width: 60px;
    font-size: 0.95rem;
}

select {
    flex: 1;
    padding: 12px 15px;
    border: 2px solid #e0e0e0;
    border-radius: 12px;
    font-size: 1rem;
    background: white;
    cursor: pointer;
    transition: all 0.3s ease;
}

select:focus {
```

```
    outline: none;
    border-color: #667eea;
    box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
}

.swap-btn {
    background: #667eea;
    color: white;
    border: none;
    width: 45px;
    height: 45px;
    border-radius: 50%;
    font-size: 1.3rem;
    cursor: pointer;
    display: flex;
    align-items: center;
    justify-content: center;
    margin: 10px auto;
    transition: all 0.3s ease;
    box-shadow: 0 4px 12px rgba(102, 126, 234, 0.3);
}

.swap-btn:hover {
    transform: rotate(180deg);
    background: #764ba2;
}

.swap-btn:active {
    transform: rotate(180deg) scale(0.95);
}

.translation-area {
    background: white;
    border-radius: 20px;
    padding: 25px;
    box-shadow: 0 10px 30px rgba(0,0,0,0.2);
    flex: 1;
    display: flex;
    flex-direction: column;
    gap: 20px;
    max-height: 500px;
}

.text-box {
    position: relative;
    flex: 1;
```

```
}

.text-label {
    font-weight: 600;
    margin-bottom: 10px;
    display: block;
    color: #667eea;
    font-size: 0.9rem;
    text-transform: uppercase;
    letter-spacing: 0.5px;
}

textarea {
    width: 100%;
    min-height: 120px;
    padding: 15px;
    border: 2px solid #e0e0e0;
    border-radius: 12px;
    font-size: 1rem;
    font-family: inherit;
    resize: vertical;
    transition: all 0.3s ease;
}

textarea:focus {
    outline: none;
    border-color: #667eea;
    box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);
}

.controls {
    display: flex;
    gap: 10px;
    margin-top: 10px;
}

.btn {
    flex: 1;
    padding: 15px 25px;
    border: none;
    border-radius: 12px;
    font-size: 1rem;
    font-weight: 600;
    cursor: pointer;
    transition: all 0.3s ease;
    display: flex;
```

```
    align-items: center;
    justify-content: center;
    gap: 8px;
    box-shadow: 0 4px 12px rgba(0,0,0,0.15);
}

.btn-primary {
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: white;
}

.btn-primary:hover {
    transform: translateY(-2px);
    box-shadow: 0 6px 20px rgba(102, 126, 234, 0.4);
}

.btn-primary:active {
    transform: translateY(0);
}

.btn-secondary {
    background: #f5f5f5;
    color: #333;
}

.btn-secondary:hover {
    background: #e0e0e0;
}

.btn:disabled {
    opacity: 0.5;
    cursor: not-allowed;
    transform: none !important;
}

.btn.recording {
    background: #ef4444;
    animation: pulse 1.5s infinite;
}

@keyframes pulse {
    0%, 100% {
        opacity: 1;
    }
    50% {
        opacity: 0.7;
```

```
    }
}

.status {
    text-align: center;
    padding: 12px;
    border-radius: 12px;
    font-size: 0.9rem;
    font-weight: 500;
    margin-top: 10px;
    opacity: 0;
    transition: opacity 0.3s ease;
}

.status.show {
    opacity: 1;
}

.status.info {
    background: #e3f2fd;
    color: #1976d2;
}

.status.success {
    background: #e8f5e9;
    color: #388e3c;
}

.status.error {
    background: #ffebee;
    color: #d32f2f;
}

.icon {
    font-size: 1.2rem;
}

.divider {
    height: 2px;
    background: linear-gradient(90deg, transparent, #667eea, transparent);
    margin: 10px 0;
}

@media (max-width: 480px) {
    body {
        padding: 15px;
```

```
            }

            h1 {
                font-size: 1.5rem;
            }

            .language-selector,
            .translation-area {
                padding: 15px;
            }

            .btn {
                padding: 12px 20px;
                font-size: 0.9rem;
            }
        }
    </style>
</head>
<body>
    <div class="container">
        <header>
            <h1>  Speech Translator</h1>
            <div class="subtitle">Speak naturally, translate instantly</div>
        </header>

        <div class="language-selector">
            <div class="language-row">
                <label for="fromLang">From:</label>
                <select id="fromLang">
                    <option value="en-US">English (US)</option>
                    <option value="en-GB">English (UK)</option>
                    <option value="es-ES">Spanish (Spain)</option>
                    <option value="es-MX">Spanish (Mexico)</option>
                    <option value="fr-FR">French</option>
                    <option value="de-DE">German</option>
                    <option value="it-IT">Italian</option>
                    <option value="pt-BR">Portuguese (Brazil)</option>
                    <option value="pt-PT">Portuguese (Portugal)</option>
                    <option value="ru-RU">Russian</option>
                    <option value="ja-JP">Japanese</option>
                    <option value="ko-KR">Korean</option>
                    <option value="zh-CN">Chinese (Simplified)</option>
                    <option value="zh-TW">Chinese (Traditional)</option>
                    <option value="ar-SA">Arabic</option>
                    <option value="hi-IN">Hindi</option>
                    <option value="nl-NL">Dutch</option>
```

```
                <option value="pl-PL">Polish</option>
                <option value="tr-TR">Turkish</option>
                <option value="sv-SE">Swedish</option>
            </select>
        </div>

        <button class="swap-btn" id="swapBtn" title="Swap languages"> </button>

        <div class="language-row">
            <label for="toLang">To:</label>
            <select id="toLang">
                <option value="es">Spanish</option>
                <option value="en">English</option>
                <option value="fr">French</option>
                <option value="de">German</option>
                <option value="it">Italian</option>
                <option value="pt">Portuguese</option>
                <option value="ru">Russian</option>
                <option value="ja">Japanese</option>
                <option value="ko">Korean</option>
                <option value="zh-CN">Chinese (Simplified)</option>
                <option value="zh-TW">Chinese (Traditional)</option>
                <option value="ar">Arabic</option>
                <option value="hi">Hindi</option>
                <option value="nl">Dutch</option>
                <option value="pl">Polish</option>
                <option value="tr">Turkish</option>
                <option value="sv">Swedish</option>
                <option value="th">Thai</option>
                <option value="vi">Vietnamese</option>
                <option value="id">Indonesian</option>
            </select>
        </div>
    </div>

    <div class="translation-area">
        <div class="text-box">
            <label class="text-label">Original Text</label>
            <textarea id="inputText" placeholder="Tap the microphone to speak or type he
            <div class="controls">
                <button class="btn btn-primary" id="micBtn">
                    <span class="icon"> </span>
                    <span id="micBtnText">Speak</span>
                </button>
                <button class="btn btn-secondary" id="clearBtn">
                    <span class="icon"> </span>
```

```
                    Clear
                </button>
            </div>
        </div>

        <div class="divider"></div>

        <div class="text-box">
            <label class="text-label">Translation</label>
            <textarea id="outputText" placeholder="Translation will appear here..." read
            <div class="controls">
                <button class="btn btn-primary" id="speakBtn" disabled>
                    <span class="icon"> </span>
                    Speak Translation
                </button>
                <button class="btn btn-secondary" id="copyBtn" disabled>
                    <span class="icon"> </span>
                    Copy
                </button>
            </div>
        </div>

        <div class="status" id="status"></div>
    </div>
</div>

<script>
    // DOM Elements
    const fromLang = document.getElementById('fromLang');
    const toLang = document.getElementById('toLang');
    const swapBtn = document.getElementById('swapBtn');
    const inputText = document.getElementById('inputText');
    const outputText = document.getElementById('outputText');
    const micBtn = document.getElementById('micBtn');
    const micBtnText = document.getElementById('micBtnText');
    const clearBtn = document.getElementById('clearBtn');
    const speakBtn = document.getElementById('speakBtn');
    const copyBtn = document.getElementById('copyBtn');
    const status = document.getElementById('status');

    // Speech Recognition
    let recognition;
    let isRecording = false;

    // Initialize Speech Recognition
    if ('webkitSpeechRecognition' in window || 'SpeechRecognition' in window) {
```

```
const SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecogni
recognition = new SpeechRecognition();
recognition.continuous = false;
recognition.interimResults = true;
recognition.maxAlternatives = 1;

recognition.onstart = () => {
    isRecording = true;
    micBtn.classList.add('recording');
    micBtnText.textContent = 'Listening...';
    showStatus('Listening... Speak now', 'info');
};

recognition.onresult = (event) => {
    let interimTranscript = '';
    let finalTranscript = '';

    for (let i = event.resultIndex; i < event.results.length; i++) {
        const transcript = event.results[i][0].transcript;
        if (event.results[i].isFinal) {
            finalTranscript += transcript + ' ';
        } else {
            interimTranscript += transcript;
        }
    }

    if (finalTranscript) {
        inputText.value = finalTranscript.trim();
        translateText();
    } else {
        inputText.value = interimTranscript;
    }
};

recognition.onerror = (event) => {
    console.error('Speech recognition error:', event.error);
    isRecording = false;
    micBtn.classList.remove('recording');
    micBtnText.textContent = 'Speak';

    if (event.error === 'not-allowed') {
        showStatus('Microphone access denied. Please enable it in settings.', 'e
    } else if (event.error === 'no-speech') {
        showStatus('No speech detected. Try again.', 'error');
    } else {
        showStatus('Error: ' + event.error, 'error');
```

```
        }
    };

    recognition.onend = () => {
        isRecording = false;
        micBtn.classList.remove('recording');
        micBtnText.textContent = 'Speak';
    };
}

// Language mapping for speech synthesis
const speechLangMap = {
    'es': 'es-ES',
    'en': 'en-US',
    'fr': 'fr-FR',
    'de': 'de-DE',
    'it': 'it-IT',
    'pt': 'pt-PT',
    'ru': 'ru-RU',
    'ja': 'ja-JP',
    'ko': 'ko-KR',
    'zh-CN': 'zh-CN',
    'zh-TW': 'zh-TW',
    'ar': 'ar-SA',
    'hi': 'hi-IN',
    'nl': 'nl-NL',
    'pl': 'pl-PL',
    'tr': 'tr-TR',
    'sv': 'sv-SE',
    'th': 'th-TH',
    'vi': 'vi-VN',
    'id': 'id-ID'
};

// Event Listeners
micBtn.addEventListener('click', () => {
    if (!recognition) {
        showStatus('Speech recognition not supported in this browser', 'error');
        return;
    }

    if (isRecording) {
        recognition.stop();
    } else {
        recognition.lang = fromLang.value;
        recognition.start();
```

```
    }
});

clearBtn.addEventListener('click', () => {
    inputText.value = '';
    outputText.value = '';
    speakBtn.disabled = true;
    copyBtn.disabled = true;
    hideStatus();
});

swapBtn.addEventListener('click', () => {
    // Swap language codes
    const fromCode = fromLang.value;
    const toCode = toLang.value;

    // Convert recognition code to translation code
    const fromCodeShort = fromCode.split('-')[0];

    // Find matching recognition code for 'to' language
    let newFromCode = 'en-US';
    for (let option of fromLang.options) {
        if (option.value.startsWith(toCode.split('-')[0])) {
            newFromCode = option.value;
            break;
        }
    }

    fromLang.value = newFromCode;
    toLang.value = fromCodeShort;

    // Swap text
    const tempText = inputText.value;
    inputText.value = outputText.value;
    outputText.value = tempText;

    if (inputText.value) {
        translateText();
    }
});

inputText.addEventListener('input', () => {
    if (inputText.value.trim()) {
        translateText();
    } else {
        outputText.value = '';
```

```
        speakBtn.disabled = true;
        copyBtn.disabled = true;
    }
});

speakBtn.addEventListener('click', () => {
    if (outputText.value) {
        speakText(outputText.value, toLang.value);
    }
});

copyBtn.addEventListener('click', async () => {
    try {
        await navigator.clipboard.writeText(outputText.value);
        showStatus('Copied to clipboard!', 'success');
    } catch (err) {
        // Fallback for older browsers
        outputText.select();
        document.execCommand('copy');
        showStatus('Copied to clipboard!', 'success');
    }
});

// Translation function using LibreTranslate API
let translationTimeout;
async function translateText() {
    clearTimeout(translationTimeout);

    const text = inputText.value.trim();
    if (!text) {
        outputText.value = '';
        speakBtn.disabled = true;
        copyBtn.disabled = true;
        return;
    }

    translationTimeout = setTimeout(async () => {
        try {
            showStatus('Translating...', 'info');

            // Using MyMemory Translation API (free, no API key required)
            const sourceLang = fromLang.value.split('-')[0];
            const targetLang = toLang.value.split('-')[0];

            const url = `https://api.mymemory.translated.net/get?q=${encodeURICompor
```

```
            const response = await fetch(url);
            const data = await response.json();

            if (data.responseStatus === 200 && data.responseData.translatedText) {
                outputText.value = data.responseData.translatedText;
                speakBtn.disabled = false;
                copyBtn.disabled = false;
                showStatus('Translation complete!', 'success');
            } else {
                throw new Error('Translation failed');
            }
        } catch (error) {
            console.error('Translation error:', error);
            showStatus('Translation failed. Please try again.', 'error');
            outputText.value = '';
            speakBtn.disabled = true;
            copyBtn.disabled = true;
        }
    }, 500); // Debounce for 500ms
}

// Text-to-Speech function
function speakText(text, lang) {
    if ('speechSynthesis' in window) {
        // Cancel any ongoing speech
        window.speechSynthesis.cancel();

        const utterance = new SpeechSynthesisUtterance(text);
        utterance.lang = speechLangMap[lang] || lang;
        utterance.rate = 0.9;
        utterance.pitch = 1;

        // Try to find a voice for the selected language
        const voices = window.speechSynthesis.getVoices();
        const voice = voices.find(v => v.lang.startsWith(utterance.lang.split('-')[(
        if (voice) {
            utterance.voice = voice;
        }

        utterance.onstart = () => {
            showStatus('Speaking...', 'info');
        };

        utterance.onend = () => {
            hideStatus();
        };
```

```
            utterance.onerror = (event) => {
                console.error('Speech synthesis error:', event);
                showStatus('Speech failed. Try again.', 'error');
            };

            window.speechSynthesis.speak(utterance);
        } else {
            showStatus('Text-to-speech not supported', 'error');
        }
    }

    // Load voices (needed for some browsers)
    if ('speechSynthesis' in window) {
        speechSynthesis.onvoiceschanged = () => {
            speechSynthesis.getVoices();
        };
    }

    // Status message functions
    function showStatus(message, type) {
        status.textContent = message;
        status.className = `status show ${type}`;
    }

    function hideStatus() {
        status.classList.remove('show');
    }

    // Auto-hide status messages after 3 seconds
    let statusTimeout;
    const originalShowStatus = showStatus;
    showStatus = (message, type) => {
        clearTimeout(statusTimeout);
        originalShowStatus(message, type);
        if (type !== 'info') {
            statusTimeout = setTimeout(hideStatus, 3000);
        }
    };

    // Check for browser support on load
    window.addEventListener('load', () => {
        if (!recognition) {
            showStatus('  Speech recognition not available. You can still type and trans
            micBtn.disabled = true;
        }
```

```
            if (!('speechSynthesis' in window)) {
                showStatus('  Text-to-speech not available in this browser.', 'error');
            }
        });
    </script>
</body>
</html>
```

Features Included:

1. **Speech-to-Text** - Real-time speech recognition with visual feedback
2. **Text Translation** - Automatic translation using free API (no key needed)
3. **Text-to-Speech** - Speak translated text in target language
4. **20+ Languages** - Major world languages supported
5. **Beautiful UI** - Modern gradient design with smooth animations
6. **Mobile Optimized** - Perfect for iPhone use
7. **Offline Capable** - Works offline after first load (except translation API)
8. **Copy to Clipboard** - Easy sharing of translations
9. **Language Swap** - Quick swap between source and target languages

How to Use:

1. **In Koder:** Save this as `translator.html`
2. **Open in browser** - It will work immediately
3. **Grant microphone permission** when prompted
4. **Speak or type** to translate
5. **Listen** to translations with the speaker button

Deployment Options:

Option 1 - GitHub Pages (Free):

- Create a GitHub account
- Create a new repository
- Upload the HTML file
- Enable GitHub Pages in settings
- Get a live URL like: `https://yourusername.github.io/translator`

Option 2 - Netlify Drop (Easiest):

- Go to netlify.com/drop
- Drag and drop the HTML file
- Get instant live URL

Option 3 - Offline/Local:

- Just keep it in Koder

- Open directly in Safari
- Works without internet (except translation needs connection)

Customization Tips:

- Change colors in the CSS gradient (line 14)
- Add more languages to the dropdowns
- Adjust button sizes for easier tapping
- Modify animation speeds

The app is **100% ready to use** right now! Let me know if you want any styling adjustments or additional features!