**Prompts**

**UI**

1. We are proposin a language learning app called ALL. Detailed description follows: "TheALL (The Assisted Language Learning) is an initiative supported by the EkStep Foundation as an attempt to reduce the impact of this digital linguistic divide and aid language learners. It is an open-source, scalable project, with its code made available as a Sunbird Building Block. The initiative is being realised through a community of volunteers and partners with a vision to improve the reading and verbal skills of language learners across all age groups. It seeks to provide learners with tools that facilitate learning at their pace, tracking progress, and providing unbiased feedback for improvement. The solutions being developed leverage the latest technologies in Automated Speech Recognition(ASR), Speech Synthesis, Natural Language Processing (NLP), and Machine Learning. The subsequent releases of the app focus on monitoring and capturing usage logs, providing a UI dashboard to view the usage metrics, and incorporating supervision/feedback by a school coordinator. Post the v0.2 release of ALL, thousands of learners have adopted the solution and feedback and improvements are being collected. The initial feedback has highlighted the success of the pilot phase, with anecdotal examples like students being encouraged by their peers' progress and themselves suggesting areas for further increments to the application. The initiative and its tools are undergoing constant upgradation and there is still a lot to uncover and accomplish. We have created a login and discovery component for the app. The discovery component of the application deals with how once a new user logs in to the app, they take a series of voice recordings where they are asked to record a sentence/word on the screen. Their competency in the language is then determined by how well they can read and speak. Since we already have the code for the backend of the discovery component, we will be focusing on styling it and making the UI look attractive.
2. This is the current code we have for the discovery component: import React, { useState, useEffect } from 'react'; import api from '../../services/api'; import Navbar from '../Navbar/Navbar'; import './Discovery.css'; const Discovery = () => { const [storyDetails, setStoryDetails] = useState([]); const [currentLineIndex, setCurrentLineIndex] = useState(0); const [sessionResult, setSessionResult] = useState(null); const [isRecording, setIsRecording] = useState(false); const [recordedAudio, setRecordedAudio] = useState(null); const [feedback, setFeedback] = useState(''); const [audioRecorder, setAudioRecorder] = useState(null); const [apiResponse, setApiResponse] = useState(''); const username = localStorage.getItem('username'); const userId = '8635444062'; // replace with actual user ID const sessionId = '86354440621701972584385'; // replace with actual session ID const subSessionId = '86354440621701972584385'; // replace with actual sub-session ID const language = 'en'; useEffect(() => { const fetchStoryDetails = async () => { const data = await api.fetchLearnerContent(userId, language); setStoryDetails(data); }; fetchStoryDetails(); }, []); const handleNextLine = () => { if (currentLineIndex < storyDetails.length - 1) { setCurrentLineIndex(currentLineIndex + 1); } }; const handleRetryLine = () => { setCurrentLineIndex(currentLineIndex); }; const handleStartRecording = () => { navigator.mediaDevices.getUserMedia({ audio: true }) .then((stream) => { const recorder = new MediaRecorder(stream); recorder.start(); setAudioRecorder(recorder); setIsRecording(true); const chunks = []; recorder.ondataavailable = (e) => { chunks.push(e.data); }; recorder.onstop = () => { const recordedBlob = new Blob(chunks, { type: 'audio/wav' }); setRecordedAudio(recordedBlob); setIsRecording(false); }; }) .catch((err) => { console.error('Error accessing microphone:', err); }); }; const handleStopRecording = () => { if (audioRecorder) { audioRecorder.stop(); } }; const handleSubmitRecording = async () => { if (recordedAudio) { const audioContent = await convertBlobToBase64(recordedAudio); const originalText = storyDetails[currentLineIndex].contentSourceData[0].text; const contentId = storyDetails[currentLineIndex].contentId; const profileData = { original\_text: originalText, audio: audioContent, user\_id: userId, session\_id: sessionId, sub\_session\_id: subSessionId, language: language, date: new Date().toISOString(), contentId: contentId, contentType: 'Char', }; try { const response = await api.updateLearnerProfile(profileData); setApiResponse(response.msg); setFeedback('Profile updated successfully.'); setSessionResult('pass'); handleNextLine(); } catch (error) { console.error('Error updating learner profile:', error); setFeedback('Error updating learner profile.'); setSessionResult('fail'); } } }; const convertBlobToBase64 = (blob) => { return new Promise((resolve, reject) => { const reader = new FileReader(); reader.readAsDataURL(blob); reader.onloadend = () => { const base64data = reader.result; resolve(base64data.split(',')[1]); }; reader.onerror = (error) => { reject(error); }; }); }; if (!storyDetails.length) { return <div>Loading...</div>; } return ( <div> <Navbar username={username} /> <div className="discovery-container"> <div className="story-content"> <p>{storyDetails[currentLineIndex].contentSourceData[0].text}</p> </div> <div className="controls"> {isRecording ? ( <button onClick={handleStopRecording}>Stop Recording</button> ) : ( <button onClick={handleStartRecording}>Start Recording</button> )} <button onClick={handleSubmitRecording} disabled={!recordedAudio || isRecording}>Submit Recording</button> </div> {feedback && <div className="feedback">{feedback}</div>} {sessionResult && <div className={`result ${sessionResult}`}>{sessionResult === 'pass' ? 'Pass' : 'Fail'}</div>} {apiResponse && <div className="api-response">{apiResponse}</div>} </div> </div> ); }; export default Discovery; We must style it according to some guidelines I will mention further.
3. The discovery.js file should be styled according to the image attached. The image is an attachment of a figma template. This is how the discovery file is expected to look. Any and all styling code must be written in the discovery.css file. Ignore any previous styling done. The styling should be as close to the attached image as possible. Some key features to note are the orange gradient background, the white card which contains the buttons and text, the bolded and large text to be read out, the purple and green buttons for starting recording/stopping recording and playing an audio of recording once it has been done, the next button which gets the next sentance/word to be read, the timer which counts down from 15 seconds and moves to the next prompt after it is elapsed, and the help button which displays some instructions for how to use the website. I understand that not all these buttons exist in the discovery.js code- feel free to edit the code to make sure these are avaliable and there is no other unnecessary information.
4. You have made some good progress building this however there are some errors we must correct. Firstly, the white card which contains all the information is too small, this card must take up more space like the figma attachment and it must be centered in the page. The text to be read out also must be made larger than it is. The next button must be positioned as it is in the image, at the bottom of the white card as opposed to next to the other buttons. Any coins should be ignored for now and any reference to the coins can be taken out. The buttons are also rounded and contain icons respective to their purposes- this should be taken into account. The timer can be increased to 30 seconds. Lastly, there should be functionality added for the next button and the help button.
5. I am getting this error: Module not found: Error: Can't resolve 'react-icons/fa' in '/Users/dhanya/Desktop/EkStep/all-code-gen/src/components/Learning'
6. There still appear to be some errors in the code that need to be corrected. For one the nav bar is cut out as it starts from the center of the page, this is not what we want. The nav bar must start from the right edge of the page so that no part of it is cut out and look abrupt. Next the white card must not be dynamically sized. It must be the same constant size and cover up at least 80% of the page- it should also be centered. All the items in the white card must be placed and spaced according to the figma image and there should be gap between the components. The purple play button must only appear after some audio has been recorded and must be functional so that it can play back the audio that has been recorded. The green button must be a mic button to record audio, and once clicked it must change into a stop button to stop the recording. After the audio has been recorded it should change into a replay button to restart taking the audio.
7. I want there to be back button inplace of the nav bar on the right of the page that takes the user back to the previous login page. Additionally, the white space on which all the information is still is far too small. I want it to take up most of the page.
8. The back button must be outside the white box fixed to the top left of the page. The next button must be in the bottom right of the white box. The timer must be in the top left of the white box and the help button must be in the top right of the white box. The white box must also be BIGGER and take up the whole page. In order to do this you can increase the size margin and padding of its components or you can use an image or a shape to create a white box of the desired size