Overview:

This project involved the development of an Android application that uses Java frameworks and libraries for Optical Character Recognition (OCR), text-to-speech synthesis, and speech recognition.
The primary goal of this application was to provide a user-friendly interface for converting text into spoken words and vice versa.

Problem Statement:

The goal of the project was to make life easier for people who have trouble reading or seeing. Sometimes, reading or seeing things can be hard for some people. This app was made to help them. It aimed to create a tool that could help users read printed text by converting it into speech and also enable them to input text through speech recognition.

Technology Stack:

Java: The project was implemented in Java, a widely used programming language for Android app development.

Android Framework: The Android framework was used for building the mobile application.

OCR Library: The application integrated an OCR library to recognize and extract text from images or printed documents.

Text-to-Speech API: Android's built-in Text-to-Speech API was utilized for converting text into speech.

Speech Recognition API: The project also incorporated speech recognition capabilities for inputting text through voice commands.

Role and Responsibilities:

I played a key role in this project, responsible for:

Researching and selecting appropriate OCR and speech recognition libraries.
Designing the user interface (UI) for the Android application.
Implementing the OCR functionality to extract text from images.

Integrating the Text-to-Speech API for converting extracted text to speech.
Incorporating speech recognition to allow users to input text by speaking.
Handling error scenarios and ensuring a smooth user experience.

Architecture:
The application followed a client-server architecture, where the client-side Android app was responsible for user interactions and integrating the OCR, text-to-speech, and speech recognition components. The OCR component extracted text from images, which was then processed by the text-to-speech module for spoken output. Speech recognition allowed users to input text through voice commands.

Key Features:

OCR Text Extraction: Users could take pictures of printed text, and the OCR component would extract and display the text.
Text-to-Speech Synthesis: The application could read the extracted text aloud using Android's Text-to-Speech API.
Speech Recognition: Users could speak text inputs, which the application would recognize and convert into written text for further use.

Challenges Faced:

Library Selection: Choosing the right OCR and speech recognition libraries that were accurate and suitable for mobile devices.

User Interface Design: Designing an intuitive and accessible UI to cater to users with visual impairments.
Integration: Ensuring smooth integration of the OCR, text-to-speech, and speech recognition components within the app.

Achievements:

The project successfully addressed the needs of users with visual impairments or reading difficulties by providing an accessible and user-friendly tool for text conversion. User feedback indicated improved accessibility to printed content.

:

Why u choose this project :

I chose this project because I wanted to create something that could genuinely make a positive impact on people's lives. When I learned about the challenges faced by individuals with visual impairments or reading difficulties, I was inspired to use my skills to help them.

Additionally, the project presented interesting technical challenges, such as integrating OCR (Optical Character Recognition) and speech recognition components, and ensuring a user-friendly interface.

Img to text

If the user selects the camera option (CAMERA_REQUEST), the app initiates a camera capture using an intent. The captured image is then cropped using the CropImage library to improve OCR (Optical Character Recognition) accuracy.

If the user selects the gallery option (GALLERY_PICTURE), the app opens the gallery for image selection. The selected image is also cropped using the CropImage library.

In cases (camera or gallery), after cropping, the code uses the Google Mobile Vision API to extract text from the cropped image. The extracted text is displayed in the mResultEt EditText, and it is also stored in the text variable.

The Bit map obtained from the cropped image is used to create a Frame for text recognition. The TextRecognizer processes the frame using recognizer.detect(frame), which detects text blocks within the image.

The recognized text is then extracted from the text blocks, concatenated into a StringBuilder, and displayed in the mResultEt EditText.

The Google Mobile Vision API,

The Google Mobile Vision API, now part of Firebase ML Kit, is a powerful set of machine learning tools provided by Google for mobile app developers. It allows developers to integrate various computer vision and machine learning capabilities into their Android and iOS applications. The Mobile Vision API offers a range of functionalities, but one of its most commonly used features is Optical Character Recognition (OCR).

OCR (Optical Character Recognition):

OCR is the process of converting printed or handwritten text within images or scanned documents into machine-readable text. The Mobile Vision API's OCR capabilities allow developers to recognize and extract text from images, making it useful for applications that involve scanning, translating, or searching for text within images.

Text Recognition: Mobile Vision's TextRecognizer class is designed for text recognition. It can detect and recognize text in various languages and fonts. The OCR functionality can recognize individual characters, words, and entire blocks of text.

Speech to text :

Android Speech Recognition API:
The primary functionality of speech recognition is implemented using Android's built-in RecognizerIntent and related classes.

RecognizerIntent is used to request speech input from the user and to specify parameters like the language model, language, and prompt.

Android UI Components:
The code utilizes various Android UI components like Spinner, ImageView, EditText, and TextView to create the user interface for selecting languages and displaying the recognized text.

ArrayAdapter:
An ArrayAdapter is used to populate the Spinner with a list of languages. This adapter is created using Android's built-in ArrayAdapter class, and the data for the adapter is loaded from the string array defined in strings.xml.

Text to speech :

Android Libraries:

android.widget.ArrayAdapter: Android widget for creating a Spinner (dropdown list) with adapter functionality.
android.widget.Button: Android widget for creating buttons.
android.widget.EditText: Android widget for creating text input fields.
android.widget.Spinner: Android widget for creating dropdown lists.
android.widget.Toast: Android widget for displaying short-duration messages.

Google ML Kit for Translation:
The code uses Google ML Kit for translation services. It imports classes and methods related to translation, such as com.google.mlkit.nl.translate.Translation, com.google.mlkit.nl.translate.Translator, and com.google.mlkit.nl.translate.TranslatorOptions. Google ML Kit provides machine learning-powered language translation capabilities.

Google Services:
The code interacts with Google services for language model downloads. This is evident in the use of com.google.mlkit.common.model.DownloadConditions, which is part of Google ML Kit

.

Text-to-Speech API:

The code utilizes the Android Text-to-Speech (TTS) API, which is part of the Android framework. It allows the app to convert text into speech.
Overall, the code combines Android's native libraries, the Text-to-Speech API, and Google ML Kit for translation to create a text-to-speech and translation application.

Explanation :