# Feature 1: Voice/Language Assessment Testing

## *Steps to implement:*

### Voice Recording and Upload Interface:

Add a front-end interface where candidates can record their voices.

Use Web APIs like the Media Recorder API (HTML5) to handle client-side voice recording.

Save the recorded audio file in a compatible format (e.g., MP3, WAV).

Allow users to upload their recorded audio, which will be stored securely in the database or cloud storage (like Azure Blob or AWS S3).

### Backend API for Audio Submission:

Create an API endpoint to handle voice recording uploads. Implement validation checks (e.g., file type, size).

Save the audio files in the file system or cloud, and link the file's metadata to the candidate's profile in the database.

### Assessment Logic:

Implement or integrate a third-party service (like a voice assessment API) to evaluate the voice recording.

Use external services such as Google Cloud Speech API or Azure Cognitive Services for language proficiency checks and scoring.

### Feedback and Scoring:

Once processed, store the score and feedback for each candidate.

Display results on the candidate's dashboard, similar to the MCQ test results.

### Admin Review Panel:

Create a section in the admin dashboard where admins can manually listen to and review the voice recordings if needed.

## *Tech Stack/Tools:*

* Web APIs for voice recording (HTML5)
* Cloud storage services (Azure/AWS)
* .NET Web API for file handling and integration with third-party services like Google Cloud Speech API.

# Feature 2: Video-based Assessment

## *Steps to implement:*

### Video Player Integration:

Add a video player on the candidate’s test page, where they can watch pre-uploaded videos.

Use HTML5 <video> elements or any modern video library like Video.js for handling different formats and providing controls.

### Questionnaire Creation:

Once the video is watched, present questions related to the content of the video.

Create a relational data model linking video content with questions and categories (similar to the MCQ test structure).

### Time-Synced Question Display (Optional):

For a more dynamic approach, sync questions with specific video timestamps. When the video reaches certain points, questions will appear based on the video section.

Implement JavaScript events to handle timed question displays.

### Answer Validation:

Similar to the existing MCQ feature, validate and score the responses after submission.

Results and feedback will be displayed on the candidate's dashboard.

## *Tech Stack/Tools:*

* Video.js or HTML5 <video> for playing videos
* .NET MVC for linking videos with corresponding questions

# Feature 3: Candidate Voice Sample Upload for Admin Review

## *Steps to implement:*

### Voice Upload Interface:

Provide candidates with an option to upload a voice sample (either recorded or pre-recorded).

Validate the file format and size before submission.

### File Storage:

Store the uploaded voice samples in a secure location (either local storage or cloud).

Link each voice sample to the candidate’s profile in the database.

### Admin Review:

Create a dedicated section in the admin dashboard where uploaded voice samples can be accessed and played.

Provide options for admins to leave feedback or notes after reviewing the voice samples.

## *Tech Stack/Tools:*

* File handling and storage in .NET Core API
* Integration with cloud storage (Azure Blob, AWS S3)
* Admin interface for voice review

# Sorting by Priority & Easiness:

## *Feature 3: Candidate Voice Sample Upload for Admin Review*

* Priority: HIGH
* Easiness: EASY

**Reason:** Allowing candidates to upload voice samples and letting admins review them is a relatively straightforward feature with clear value. It involves basic file uploads and admin management.

## *Feature 2: Video-based Assessment*

* Priority: MEDIUM
* Easiness: MEDIUM

**Reason:** Integrating video playback is slightly more complex due to handling media files and synchronizing questions with video. However, it still has significant value for assessing candidate comprehension and is manageable using HTML5 and backend integration.

## *Feature 1: Voice/Language Assessment Testing*

* Priority: HIGH
* Easiness: HARD

**Reason:** While crucial for testing real-world language skills, it involves more complexity due to voice recording, external API integration (e.g., for voice analysis), and real-time feedback mechanisms. This feature has significant value but requires more time and development effort due to third-party API integration and audio handling.