# **TajweedAI: A Technical Roadmap for Quranic Recitation Mistake Detection Using NVIDIA Riva and NeMo**

## **Introduction**

The precise and melodious recitation of the Holy Quran, governed by the intricate science of Tajweed, represents a cornerstone of Islamic devotional practice.1 The traditional method for mastering this art is

*Talaqqi wa Mushafahah*—a direct, face-to-face learning process with a qualified teacher who provides immediate, personalized correction.3 While this remains the gold standard, its inherent limitations in scalability and accessibility have created a significant need for technological aids that can support learners in their journey.3 Existing digital tools often focus on high-level functionalities like memorization assistance or basic word-error detection, leaving a critical gap in providing the granular, phonetic-level feedback required for true mastery of Tajweed.5

This document presents a comprehensive technical roadmap for the development of **TajweedAI**, a system conceived to address this gap. TajweedAI is envisioned as an advanced Quranic recitation mistake detection system that moves beyond simple transcription accuracy to analyze the phonetic correctness of specific Tajweed rules. The project's scope is strategically focused on two foundational yet acoustically distinct rules: **Madd** (vowel elongation) and **Qalqalah** (the resonant articulation of specific plosive consonants). This focused approach allows for the development of specialized, high-accuracy models while establishing a framework that is extensible to other rules in the future.

The core of this roadmap is built upon a foundation of feasibility for an individual developer operating on a standard personal computer. This constraint dictates a pragmatic and resource-aware approach, leveraging the formidable capabilities of the NVIDIA AI ecosystem—specifically, the **NVIDIA NeMo** framework for model development and the **NVIDIA Riva** SDK for efficient, local deployment.7 The data pipeline is designed to harness the rich, open-source assets of Tarteel's

**Quranic Universal Library (QUL)**, a comprehensive collection of Quranic audio and textual resources.9

The objective of this report is to furnish an exhaustive, expert-level blueprint that is both scientifically grounded and practically actionable. It will navigate the entire project lifecycle, commencing with a deep analysis of the acoustic-phonetic principles that underpin the target rules. It will then detail a novel data processing pipeline for creating a ground-truth dataset, followed by a phased plan for model development and training. Finally, it will propose a robust, scalable system architecture for a full-stack web application, providing a long-term vision for the project's growth and impact. This document serves as a definitive guide for transforming the concept of TajweedAI into a tangible, high-impact tool for Quranic learners worldwide.

## **Section 1: An Acoustic-Phonetic Framework for Tajweed Rule Detection**

The development of any successful phonetic event detection system must begin not with code, but with a rigorous scientific understanding of the target phenomena. A machine learning model can only learn to identify patterns that are well-defined and measurable. Therefore, this section deconstructs the traditional, descriptive definitions of Madd and Qalqalah into a concrete framework of acoustic-phonetic correlates. This framework translates the art of Tajweed into the language of signal processing, establishing the quantitative "ground truth" upon which the entire TajweedAI system will be built.

### **1.1 The Durational Characteristics of Madd (Vowel Elongation)**

At its core, the rule of Madd is a prescription for controlling vowel duration.11 Its correct application is fundamental to the rhythm and melodic flow of Quranic recitation, and its misapplication can, in some cases, alter the meaning of words.13 The primary and most dominant acoustic correlate for detecting Madd is, therefore, the temporal length of the vowel sound.

The rule applies specifically to the three long vowels in the Arabic language, known as the *Huruf al-Madd* (letters of Madd). These occur under specific contextual conditions: the letter Alif (ا) when preceded by a character with a Fatha vowel, the letter Yaa' (ي) when preceded by a character with a Kasra vowel, and the letter Waaw (و) when preceded by a character with a Dammah vowel.12

The unit of measurement for this elongation is the *harakah* (plural: *harakat*), which translates to "a movement." A *harakah* is a relative unit of time, defined as the duration required to pronounce a single short vowel (like 'ba', 'bi', or 'bu').14 Consequently, the absolute duration of a

*harakah* in milliseconds is not fixed; it is dependent on the overall tempo of the recitation. A slower recitation will have a longer *harakah* duration than a faster one.16 This relativity is a critical consideration for the model; a fixed millisecond threshold for correctness would fail across different recitation speeds. Academic analysis of a reference reciter, Sheikh El-Hosary, provides a useful quantitative baseline, finding the mean duration of one "movement time" to be approximately 379 milliseconds in his particular style.17

The various types of Madd are distinguished by their prescribed duration in *harakat*:

* **Madd Asli / Tabi'i (Natural Madd):** This is the most fundamental and frequent form of elongation, occurring in the absence of any subsequent trigger like a Hamza (ء) or Sukoon (ْ). Its duration is strictly **2 harakat**.11 It is the baseline against which other Madd rules are measured.
* **Madd Far'i (Secondary Madd):** This category encompasses all elongations that extend beyond the natural 2 harakat, triggered by the presence of a Hamza or Sukoon following the letter of Madd.11 The key sub-types include:
  + **Madd Muttasil (Connected Madd):** This is an obligatory elongation of **4 or 5 harakat**. It occurs when a letter of Madd is followed by a Hamza within the same word.11
  + **Madd Munfasil (Separated Madd):** This is a permissible elongation of **4 or 5 harakat**, though it can be shortened to 2 harakat depending on the recitation tradition (*riwayah*). It occurs when a letter of Madd is at the end of one word and the following word begins with a Hamza.11
  + **Madd 'Aridh li-l-Sukoon (Temporary Madd for a Stop):** This elongation is flexible, allowing for **2, 4, or 6 harakat**. It occurs when a reciter pauses on a word where the letter following the Madd letter takes on a temporary Sukoon due to the pause.11
  + **Madd Lazim (Necessary Madd):** This is the longest and most emphatic elongation, with a mandatory duration of **6 harakat**. It is triggered when a letter of Madd is followed by a character with a permanent Sukoon or a Shaddah (a mark indicating a doubled consonant).11

From a detection standpoint, the task for Madd is clear: it is a problem of accurately measuring vowel duration and classifying that duration into the correct *harakah* category, relative to the reciter's tempo. This can be framed either as a regression problem (predicting the duration in milliseconds or as a ratio) or, more practically, as a multi-class classification problem (e.g., "Correct 2 harakat," "Incorrect - Too Short," etc.).

### **1.2 The Multi-Feature Signature of Qalqalah (Plosive Consonant Echo)**

Unlike the primarily one-dimensional nature of Madd, Qalqalah is a complex, multi-faceted phonetic event. Traditionally described as a "shaking," "disturbance," or "echo," it applies to five specific voiceless plosive consonants—ق, ط, ب, ج, د (often remembered by the mnemonic phrase *qutbu jadd*)—when they carry a Sukoon (a diacritic indicating the absence of a vowel).19 Phonetically, Qalqalah is the audible and resonant release phase of these stop consonants. This release prevents the sound from being abruptly cut off, which would otherwise happen due to the combination of their articulatory properties (

*jahr*, stoppage of breath flow, and *shiddah*, stoppage of sound flow).19

While the "echo" metaphor is useful for learners, a robust detection system requires a more precise, quantitative definition. Recent acoustic phonetic research provides exactly this, moving beyond traditional descriptions to identify a set of measurable acoustic correlates. A 2024 study on the acoustics of Qalqalah reveals that the event is characterized by distinct and measurable changes in several features 21:

* **Duration:** The inserted vowel-like sound that constitutes the Qalqalah release has a measurable duration.
* **Intensity (Amplitude):** The release burst has a specific energy profile, making it louder than if the sound were simply cut off.
* **Fundamental Frequency (F0 / Pitch):** The pitch contour of the voice is measurably perturbed during the Qalqalah event.
* **Formant Frequencies (F1 and F2):** The spectral shape of the sound, captured by the formant frequencies, shows a clear transition. The study indicates that the inserted sound is a central vowel whose quality is influenced by the surrounding phonetic context.21

Furthermore, this research provides a crucial clarification regarding the different types of Qalqalah. Traditionally, these are categorized by perceived strength based on their position within a word or at a pause: *Qalqalah Sughra* (Minor/Subtle), *Qalqalah Wusta* (Medium), and *Qalqalah Kubra* (Major/Strong).19 However, the acoustic evidence presents a counter-intuitive but vital finding:

**Minor Qalqalah (*Sughra*) is acoustically more prominent than Major Qalqalah (*Kubra*)**.21

* **Minor Qalqalah (*Sughra*):** Occurring in the middle of a word or at the end of a word when not pausing, this is identified as a **phonological process** akin to vowel epenthesis. A full vowel is effectively inserted to break up a difficult consonant cluster. Acoustically, this results in a *longer duration*, *higher intensity*, and *higher F0* compared to Major Qalqalah.21
* **Major Qalqalah (*Kubra*):** Occurring at the end of a word when pausing, this is identified as a **phonetic process** that represents the simple, mechanical release phase of the stop consonant. Acoustically, this makes it *less prominent*, with a shorter duration, lower intensity, and a falling F0 contour typical of the end of an utterance.21

This distinction is paramount for the design of TajweedAI. A model trained on the traditional (and acoustically inaccurate) assumption that Major Qalqalah should be "stronger" would be fundamentally flawed. The detection strategy for Qalqalah must therefore be a multi-dimensional pattern recognition task. It requires a model capable of learning the complex, non-linear interplay between duration, intensity, pitch, and spectral features (like MFCCs, which encapsulate formant information). This complexity makes it an ideal candidate for a deep learning classifier, such as a Convolutional Neural Network (CNN), which excels at identifying such patterns in spectrogram-like data representations.22

**Table 1: Acoustic Correlates of Target Tajweed Rules**

This table serves as a foundational reference for the project, translating the abstract rules of Tajweed into concrete, measurable engineering targets. It distills the findings from this section into an actionable format for the developer, bridging the gap between Islamic phonetics and signal processing. This clarifies that Madd is primarily a one-dimensional problem (duration) while Qalqalah is a multi-dimensional one, a distinction that directly influences the choice of feature extraction methods and model complexity detailed in Section 3.

| Tajweed Rule | Primary Acoustic Correlate | Key Measurable Features | Expected Quantitative Signature (Reference) | Relevant Research |
| --- | --- | --- | --- | --- |
| **Madd Asli (Natural)** | Vowel Duration | duration\_ms, relative\_duration\_ratio | 2 *harakat* (approx. 2x reciter's avg. short vowel duration) | 11 |
| **Madd Muttasil** | Vowel Duration | duration\_ms, relative\_duration\_ratio | 4-5 *harakat* | 11 |
| **Madd Munfasil** | Vowel Duration | duration\_ms, relative\_duration\_ratio | 4-5 *harakat* (permissible 2) | 11 |
| **Madd Lazim** | Vowel Duration | duration\_ms, relative\_duration\_ratio | 6 *harakat* (obligatory) | 11 |
| **Qalqalah Sughra (Minor)** | Plosive Release Burst | duration\_ms, intensity\_dB, F0\_contour, MFCC\_delta | Acoustically prominent: longer duration, higher intensity & F0 vs. Major. A phonological vowel insertion. | 21 |
| **Qalqalah Kubra (Major)** | Plosive Release Burst | duration\_ms, intensity\_dB, F0\_contour, MFCC\_delta | Acoustically less prominent: shorter duration, lower intensity & F0. A phonetic release phase. | 21 |

## **Section 2: The Data Ecosystem: Leveraging Tarteel and Public Annotations**

The success of any supervised machine learning project is contingent upon the quality and availability of its training data. This section outlines a strategic approach to constructing the necessary dataset for TajweedAI by synergistically combining the rich resources of Tarteel's Quranic Universal Library (QUL) with publicly available textual annotations. This data-centric strategy is the cornerstone of the project's feasibility.

### **2.1 Analysis of the Tarteel Platform and the QUL Initiative**

Tarteel has established itself as a leading AI-powered application for Quranic learning, primarily focused on memorization and recitation practice.4 Its most prominent feature, from a technical perspective, is its real-time mistake detection. As a user recites, the application follows along, highlighting words on the screen and flagging insertions, deletions, or substitutions.5 This functionality demonstrates a sophisticated underlying Automatic Speech Recognition (ASR) engine.

However, an analysis of user feedback and the platform's stated features reveals a significant opportunity. Users on platforms like Reddit explicitly express a desire for more granular feedback that goes beyond simple word accuracy, specifically requesting checks for Tajweed rules like the length of *Madd* or the application of *Ghunnah* (nasalization).6 This identified gap in the market validates the core value proposition of TajweedAI: to provide the phonetic-level feedback that current tools lack.

The key to unlocking this project's potential lies in Tarteel's open-source data initiative, the **Quranic Universal Library (QUL)**. QUL is a centralized, curated hub of digital Quranic resources designed to empower developers.9 While Tarteel understandably keeps its proprietary AI models and user recitation datasets private to maintain its sustainability 10, the resources it has open-sourced are immensely valuable.

For TajweedAI, the most critical assets within QUL are:

* **Recitations and Segments Data:** QUL provides a collection of high-quality audio recordings from renowned Quran reciters (e.g., Sheikh El-Hosary, a common reference in academic studies 1). Crucially, many of these recitations come with detailed, pre-existing  
  **word-by-word and ayah-by-ayah timestamp data**.9 This resource provides the raw audio material and a preliminary alignment, significantly reducing the initial data processing workload.
* **Quran Script Data:** The library contains standardized Quranic text in multiple scriptural variants (e.g., Uthmani, Imlaei), available for download in developer-friendly formats like JSON and SQLite.25 This provides the pristine, ground-truth text necessary for accurate ASR and alignment.
* **Tajweed Rules Annotation Tool:** QUL features a web-based tool for community members to review, correct, and annotate Tajweed rules that are embedded within the Quranic text.9 While a fully annotated dataset is not offered as a direct download 27, the existence of this tool and its underlying data schema 28 confirms that Tarteel possesses a structured, character-level representation of Tajweed rules. This provides a vital blueprint for how we can structure our own annotations.

A pivotal discovery connects the Tarteel ecosystem directly to the project's mandated technology stack. The user query requires the use of NVIDIA NeMo. Initially, the unavailability of Tarteel's models seems to pose a challenge. However, a close examination of the model card for NVIDIA's premier pre-trained Arabic ASR model, stt\_ar\_fastconformer\_hybrid\_large\_pcd\_v1.0, reveals a game-changing fact: it was trained on a composite dataset that includes **390 hours of audio from the "TarteelAI Everyayah" dataset**.30

This symbiotic relationship between Tarteel's data and NVIDIA's models is the single most important factor ensuring the feasibility of TajweedAI for a solo developer. It means that a state-of-the-art, off-the-shelf NVIDIA model is already highly optimized for the specific acoustic domain of Quranic recitation, including the very reciters and recording conditions present in the QUL datasets. This dramatically lowers the barrier to entry. The project is no longer about the monumental task of building a world-class Arabic ASR system from scratch; instead, it becomes the much more manageable task of building specialized phonetic classifiers on top of a powerful, domain-adapted feature extractor.

### **2.2 Survey of External Textual Annotation Resources**

While QUL provides the audio and text, another crucial component is needed: a comprehensive, machine-readable map of where every Tajweed rule occurs in the Quranic text. The cpfair/quran-tajweed GitHub repository provides precisely this resource.31

This public repository contains a file, tajweed.hafs.uthmani-pause-sajdah.json, which serves as a complete set of Tajweed annotations for the entire Quran according to the *Hafs 'an 'Asim* recitation, the most common recitation style worldwide.32 The data is structured as an array of objects, where each object corresponds to an ayah of the Quran. Within each ayah object, an

annotations array lists every applicable Tajweed rule. Each rule annotation contains three key fields 31:

* "rule": A string identifying the specific rule (e.g., "madd\_6" for a 6-harakah Madd, "qalqalah" for Qalqalah).
* "start": An integer representing the starting character index of the rule.
* "end": An integer representing the ending character index of the rule.

This file provides the "textual ground truth" for the project. It allows us to know, with character-level precision, where a specific rule should be applied in the reference text.

A critical implementation detail must be strictly observed: the start and end indices in this JSON file are Unicode codepoint offsets that correspond to a **specific, historical version of the Tanzil.net Uthmani Quran text file (quran-uthmani.txt) from circa April 6, 2017**.31 Using any other version of the Quranic text will result in misaligned annotations, as even minor changes in encoding or the inclusion of pause marks can shift the character indices. Therefore, it is imperative to source and use this exact text file as the textual basis for the entire data pipeline. Attempting to rebuild the annotations for a different text file is a significant undertaking that falls outside the scope of this project.

## **Section 3: The TajweedAI Technical Roadmap: A Phased Implementation Plan**

This section presents the core of the roadmap: an actionable, step-by-step guide for constructing the TajweedAI system. The plan is divided into two primary phases. Phase 1 focuses on the critical task of data engineering: creating a unified, multi-modal dataset that links audio segments with their corresponding Tajweed rule labels. Phase 2 details the development and training of the machine learning models that will perform the actual mistake detection.

### **Phase 1: Ground-Truth Dataset Construction**

The foundational challenge of this project is the creation of a high-quality, ground-truth dataset. We possess audio from QUL and textual rule locations from an external repository, but these exist in separate domains. The audio is time-based, while the rule annotations are character-based. The central innovation of this phase is a novel data fusion pipeline designed to bridge this gap automatically, creating a single, coherent dataset ready for model training.

#### **Step 3.1.1: Data Acquisition and Preparation**

The first step involves gathering and organizing all the necessary raw materials. This process must be executed with meticulous attention to detail to ensure compatibility between the different data sources.

1. **Download Audio Recitations:** From the Tarteel QUL repository, download the high-quality, segmented audio recordings of a reference reciter. Sheikh Mahmoud Khalil Al-Hosary is an excellent choice, as his recitation is widely considered a benchmark and is used in several academic studies on automated Tajweed analysis.1 Focus on recordings that already have associated word-level timestamp data, if available, to serve as a validation source later.
2. **Acquire the Correct Quranic Text:** Locate and download the specific version of the quran-uthmani.txt file (dated circa April 6, 2017) that the cpfair/quran-tajweed annotations are based on.31 This is a non-negotiable prerequisite for accurate alignment.
3. **Load Textual Annotations:** Parse the tajweed.hafs.uthmani-pause-sajdah.json file from the cpfair/quran-tajweed repository into memory using a standard JSON library in Python. This will create a data structure that maps each surah and ayah to a list of character-indexed Tajweed rules.31

#### **Step 3.1.2: Generating Word-Level Timestamps with NeMo Forced Aligner (NFA)**

With the audio and its corresponding ground-truth text, the next step is to determine the precise start and end times of every single word in the audio recordings. The NVIDIA NeMo Forced Aligner (NFA) is the ideal tool for this task, offering high accuracy by leveraging a powerful, pre-trained ASR model.34

The process for generating these timestamps is as follows:

1. **Prepare the NeMo Manifest:** Create a JSON manifest file (e.g., qul\_manifest.json). This file is the primary input for NFA. Each line of the file will be a separate JSON object representing one audio segment (e.g., one ayah). Each object must contain two key-value pairs:
   * "audio\_filepath": The absolute path to the audio file (e.g., /path/to/qul\_audio/surah\_1\_ayah\_1.wav).
   * "text": The corresponding ground-truth text for that ayah, taken directly from the specific quran-uthmani.txt file acquired in the previous step.
2. **Execute the NFA Script:** Run the align.py script provided within the NeMo toolkit's tools directory. The command will be structured as follows 34:  
   Bash  
   python <path\_to\_NeMo>/tools/nemo\_forced\_aligner/align.py \  
    manifest\_filepath="/path/to/qul\_manifest.json" \  
    pretrained\_name="nvidia/stt\_ar\_fastconformer\_hybrid\_large\_pcd\_v1.0" \  
    output\_dir="/path/to/nfa\_output/"  
   * pretrained\_name: This parameter is crucial. It specifies the NVIDIA Arabic ASR model to use for generating the alignments. The stt\_ar\_fastconformer\_hybrid\_large\_pcd\_v1.0 model is the optimal choice because it is a state-of-the-art Conformer model that supports diacritics and was trained on Tarteel's own data, ensuring high relevance and accuracy for this specific task.30 NFA requires a CTC-based model, and this hybrid model can operate in CTC mode.
3. **Process the Output:** For each audio file listed in the manifest, NFA will generate a corresponding .ctm (CTM) file in the specified output directory (under ctm/words/). This file contains the word-level alignments. Each line in the CTM file represents one word and follows the format: <utterance\_id> 1 <start\_time\_seconds> <duration\_seconds> <word>.34

#### **Step 3.1.3: Fusing Textual Annotations with Audio Timestamps**

This is the final and most innovative step of the data construction phase. It involves programmatically linking the character-based rule annotations to the time-based word annotations. A custom Python script will be required to execute the following algorithm:

1. **Initialize Data Structures:** Create a data structure (e.g., a list of dictionaries) to hold the final, fused annotation data.
2. **Iterate Through Ayahs:** Loop through each ayah for which you have both an audio file and a parsed textual annotation from cpfair/quran-tajweed.
3. **Load Word Timestamps:** For the current ayah, open and parse its corresponding word-level CTM file generated by NFA.
4. **Track Character Position:** Initialize a character pointer to 0 for the ground-truth text of the current ayah.
5. Map Words to Rules: For each word and its start\_time and duration from the CTM file:  
   a. Determine the character span of the current word in the ground-truth text. The starting character index is the current value of the character pointer. The ending index is start\_index + len(word).  
   b. Iterate through the list of pre-loaded Tajweed rule annotations for the current ayah.  
   c. For each rule, check if the rule's character span (rule.start to rule.end) overlaps with the current word's character span.  
   d. If there is an overlap, a link is established. Create an annotation object containing the word's text, its start and end timestamps, and the details of the associated Tajweed rule (e.g., rule\_name: "madd\_6").  
   e. Append this fused annotation object to your final data structure.  
   f. Update the character pointer by advancing it past the current word and any subsequent whitespace.
6. **Persist the Fused Dataset:** After processing all ayahs, save the final data structure. The recommended format is a MongoDB collection, as its flexible document structure is ideal for this type of nested data. A detailed schema is proposed in Section 4.

At the conclusion of this phase, the developer will possess a powerful, custom-built dataset where each entry represents a specific word from a recitation, complete with its precise audio timestamps and a list of any applicable Madd or Qalqalah rules. This dataset is the fuel for the model development in Phase 2.

**Table 2: Data Sources and Tools for the TajweedAI Pipeline**

This table provides a clear, one-glance summary of all required assets and software for the data construction phase. It serves as a checklist for the developer, consolidating scattered information and highlighting critical dependencies to prevent common setup errors.

| Component | Source/Tool | Key Identifier/Version | Role in Pipeline |
| --- | --- | --- | --- |
| **Recitation Audio** | Tarteel Quranic Universal Library (QUL) | Reciter: Sheikh Mahmoud Al-Hosary (or other high-quality reference) | Raw audio input for analysis and alignment. |
| **Quranic Text** | cpfair/quran-tajweed GitHub Repo | quran-uthmani.txt (ca. Apr 6, 2017) | Ground-truth text for forced alignment; reference for character indices. |
| **Textual Tajweed Annotations** | cpfair/quran-tajweed GitHub Repo | tajweed.hafs.uthmani-pause-sajdah.json | Source of character-level locations for all Tajweed rules. |
| **Word Timestamps** | NVIDIA NeMo Forced Aligner (NFA) | NeMo Toolkit (e.g., v1.23.0+) | Generates precise word-level start and end times for audio files. |
| **ASR Model for Alignment** | NVIDIA NGC / Hugging Face | nvidia/stt\_ar\_fastconformer\_hybrid\_large\_pcd\_v1.0 | The engine within NFA that provides the phonetic intelligence for alignment. |

### **Phase 2: Phonetic Event Detection Model Development**

With the unified, time-aligned, and rule-annotated dataset from Phase 1, the project now shifts to the core machine learning task: training models to automatically detect correct and incorrect applications of Madd and Qalqalah. A pragmatic, hybrid modeling strategy is proposed. This approach tailors the model architecture to the specific acoustic complexity of each rule, a decision that enhances both performance and feasibility for a solo developer working on a personal computer. This avoids the high computational cost and complexity of a single, monolithic deep learning model for all tasks.36

#### **Step 3.2.1: Acoustic Feature Extraction**

For each word in our fused dataset that is annotated with a Madd or Qalqalah rule, the corresponding audio segment must be extracted using its start and end timestamps. From this segment, we will extract a set of numerical features that capture the acoustic essence of the rule.

* Feature Extraction for Madd:  
  As established in Section 1, Madd is primarily a durational phenomenon.
  1. **Vowel Duration Measurement:** The first step is to precisely measure the duration of the vowel within the word's audio segment. This can be accomplished using phonetic analysis tools like Praat (via scripting) or by implementing a vowel detection algorithm in Python based on acoustic cues like high energy and stable formant structures.38
  2. **Tempo Normalization:** Raw duration in milliseconds is not a robust feature, as it varies with recitation speed. The key feature will be a **relative duration ratio**. This is calculated by normalizing the measured Madd vowel duration against the reciter's local tempo. A simple and effective method is to find a nearby short vowel in the same utterance, measure its duration, and compute the ratio: relative\_duration=durationmadd​/durationshort\_vowel​. A correctly recited 2-harakah Madd should yield a ratio of approximately 2.0. This makes the feature invariant to the speed of recitation.
* Feature Extraction for Qalqalah:  
  Qalqalah is a more complex event requiring a richer feature set to capture its multi-dimensional acoustic signature.
  1. **Mel-Frequency Cepstral Coefficients (MFCCs):** From the audio segment corresponding to the Qalqalah consonant's release, extract a sequence of MFCC vectors. MFCCs are the industry standard for representing the spectral envelope (shape) of a sound and are highly effective in speech recognition and phonetic classification tasks.3
  2. **Prosodic Features:** Extract the **pitch (F0) contour** and the **intensity (energy) contour** over the same audio segment. These capture the melodic and dynamic aspects of the Qalqalah release burst.
  3. **Feature Aggregation:** Concatenate the MFCCs, pitch, and intensity features over time to create a 2D feature map, similar to a spectrogram. This 2D representation captures both the spectral and temporal evolution of the Qalqalah event and is an ideal input for a Convolutional Neural Network.

#### **Step 3.2.2: Model Selection and Implementation**

The choice of model architecture is tailored to the complexity of the features extracted for each rule.

* **Model for Madd (Classification):**
  + **Proposed Model:** A classical Machine Learning classifier, specifically a **Support Vector Machine (SVM)** with a radial basis function (RBF) kernel.22
  + **Task:** The SVM will be trained to perform multi-class classification. Its input will be the single relative\_duration\_ratio feature. Its output will be one of several classes: "Correct 2 harakat," "Correct 4-5 harakat," "Correct 6 harakat," "Incorrect (Too Short)," or "Incorrect (Too Long)."
  + **Rationale:** The classification problem for Madd is low-dimensional, relying on a single, powerful feature. An SVM is computationally inexpensive to train, highly effective at finding optimal decision boundaries in such spaces, and more interpretable than a deep learning model, making it a perfect fit for this task and the project's feasibility constraints.36
* **Model for Qalqalah (Classification):**
  + **Proposed Model:** A lightweight, custom **2D Convolutional Neural Network (CNN)**.
  + **Task:** The CNN will be trained as a binary classifier. Its input will be the 2D feature map (MFCCs, pitch, intensity) representing the Qalqalah event. Its output will be one of two classes: "Correct Qalqalah" or "Incorrect/Missing Qalqalah."
  + **Rationale:** A CNN is exceptionally well-suited for learning the local spatio-temporal patterns present in spectrogram-like data.22 It can automatically learn to detect the characteristic shape of the Qalqalah release burst in the combined feature space, capturing the complex interplay of spectral content, pitch, and energy that defines the rule. A small, custom CNN architecture (e.g., a few convolutional layers followed by pooling and dense layers) can be effectively trained on a standard personal computer with a modern GPU, staying within the project's hardware constraints.

#### **Step 3.2.3: Training, Evaluation, and Iteration Strategy**

The final step is to train and validate these models.

* **Generating Training Data:**
  + **"Correct" Examples:** The features extracted from the reference reciter's audio (e.g., Sheikh El-Hosary) will serve as the positive examples for the "Correct" classes.
  + **"Incorrect" Examples:** Generating negative examples is a common challenge. Several strategies can be employed:
    1. **Use Learner Data:** If available, recordings from novice Quran learners can provide a rich source of genuine mistakes.
    2. **Data Augmentation/Synthesis:** Programmatically create incorrect examples by altering the correct ones. For Madd, this could involve artificially shortening or lengthening the vowel segment. For Qalqalah, the release burst could be attenuated or removed entirely from the audio signal before feature extraction.
* **Training and Evaluation:**
  + Divide the constructed dataset into standard training, validation, and test splits (e.g., 80/10/10).
  + Train the SVM and CNN models on the training set, using the validation set to tune hyperparameters (e.g., the C and gamma parameters for the SVM, or the learning rate and number of filters for the CNN).
  + Evaluate the final performance of the trained models on the unseen test set using standard classification metrics: **Accuracy, Precision, Recall, and F1-Score**. For the Madd model, Mean Absolute Error (MAE) on the predicted duration ratio can also provide valuable insight.
* **Iterative Refinement:** The process of model development is cyclical. An analysis of the models' failures on the validation set (i.e., looking at the confusion matrix and specific misclassified examples) can provide direction for improvement. This could involve refining the feature extraction process, adjusting the model architecture, or augmenting the dataset with more challenging examples. Some systems incorporate a human-in-the-loop process where an expert reviews and corrects model predictions, which are then fed back into the training set to periodically refine the model.41 While a dedicated expert is beyond the scope of a solo project, the developer can perform this role on a smaller scale to incrementally improve model accuracy.

**Table 3: Proposed Model Architectures for Rule Detection**

This table explicitly defines the recommended modeling approach for each target rule, justifying the choice based on acoustic complexity and the project's feasibility constraints. It operationalizes the hybrid modeling strategy, providing a clear and distinct plan for Madd and Qalqalah and preventing the developer from defaulting to a single, overly complex model that might be suboptimal and difficult to train.

| Tajweed Rule | ML Task | Proposed Model | Input Features | Output Labels | Rationale |
| --- | --- | --- | --- | --- | --- |
| **Madd** | Multi-Class Classification | Support Vector Machine (SVM) | 1D Feature: relative\_duration\_ratio | Correct (2, 4-5, 6 harakat), Incorrect (Too Short/Long) | Low-dimensional feature space; computationally efficient and highly interpretable. |
| **Qalqalah** | Binary Classification | 2D Convolutional Neural Network (CNN) | 2D Feature Map: MFCCs + Pitch + Intensity Contours | Correct, Incorrect/Missing | Complex spatio-temporal pattern; CNNs excel at learning from spectrogram-like inputs. |

## **Section 4: System Architecture and Long-Term Vision**

With trained and validated models for Madd and Qalqalah detection, the final stage of the project involves assembling these components into a functional, user-facing application. This section outlines a comprehensive architecture, covering local deployment for development and testing, the design of a full-stack web application for user interaction, and a forward-looking vision for the future evolution of TajweedAI.

### **4.1 Local Deployment and Inference Pipeline with NVIDIA Riva**

NVIDIA Riva is a GPU-accelerated SDK designed for deploying high-performance speech AI services.7 While it is capable of powering large-scale enterprise applications, its architecture is flexible enough to be deployed locally on a developer's machine using Docker. This makes it the perfect tool for building and testing the TajweedAI inference pipeline without incurring cloud computing costs.42

The setup process for a local Riva instance is straightforward:

1. **Prerequisites:** Install Docker, the NVIDIA Container Toolkit, and obtain an NGC (NVIDIA GPU Cloud) API key.44
2. **Download Quick Start Scripts:** From the NGC catalog, download the Riva Quick Start package, which contains the necessary scripts for initialization and deployment.43
3. **Configuration:** Modify the config.sh file within the Quick Start directory. This file allows you to specify which services to enable (ASR, TTS, etc.) and which models to deploy. Here, you will specify the pre-trained Arabic ASR model: nvidia/stt\_ar\_fastconformer\_hybrid\_large\_pcd\_v1.0.
4. **Launch Server:** Execute the riva\_init.sh script to download the container images and models. Once initialization is complete, run riva\_start.sh to launch the Riva server. This command spins up a set of Docker containers that host the speech AI services, accessible via a gRPC API on a local port (typically localhost:50051).43

The core of TajweedAI's real-time analysis is not a single model call but a **multi-stage inference pipeline** orchestrated by our backend application:

* **Stage 1 (ASR and Timestamp Generation):** The client application streams raw audio to the backend, which in turn streams it to the Riva server's ASR service. The Riva ASR service performs real-time transcription and, crucially, provides word-level timestamps for the recognized text.45
* **Stage 2 (Rule Identification):** As the transcribed words are received by the backend, they are cross-referenced with the pre-computed textual annotation data (derived from cpfair/quran-tajweed in Phase 1). This step quickly identifies which words in the live recitation *should* contain a Madd or Qalqalah rule.
* **Stage 3 (Phonetic Analysis):** For each word identified in Stage 2, the backend uses its timestamps to extract the corresponding audio segment from the incoming stream. This segment is then passed as input to the appropriate custom-trained model: the Madd SVM or the Qalqalah CNN.
* **Stage 4 (Feedback Generation):** The output from the custom model (e.g., "Correct," "Incorrect: Madd too short") is packaged into a structured message, along with the word itself and its timestamps, and is prepared to be sent back to the client.

### **4.2 Full-Stack Web Application Architecture**

To deliver an interactive and intuitive user experience, a modern full-stack web application is required. The architecture must be designed from the ground up to support real-time, low-latency communication, as feedback must be delivered to the user almost instantaneously as they recite.

A standard HTTP request-response architecture is ill-suited for this task. The continuous nature of speech necessitates a persistent, bidirectional communication channel between the client and server. **WebSockets** are the industry-standard technology for this purpose, enabling the server to push analysis results to the client as soon as they are generated, without the client needing to poll for updates.47

* **Frontend (React):**
  + **Technology:** React is a robust choice for building a dynamic and responsive user interface.
  + **Audio Capture:** The browser's built-in navigator.mediaDevices.getUserMedia() API will be used to access the user's microphone and capture the audio stream.49
  + **Real-time Communication:** Upon starting a recitation session, the React application will establish a WebSocket connection to the backend server. The captured audio will be encoded, chunked into small buffers, and sent over this WebSocket.47
  + **User Interface (UI):** The UI will prominently display the Quranic text being recited. As the backend streams back analysis results (e.g., {word: "الضَّالِّينَ", rule: "Madd Lazim", status: "Correct"}), the frontend will use React's state management to dynamically update the view. This could involve color-coding the word (e.g., green for correct, red for incorrect), displaying a tooltip with specific feedback, or updating a running scorecard. While libraries like react-speech-recognition offer convenient hooks, they are typically designed for browser-native speech APIs; for a custom backend like ours, a direct implementation using the native WebSocket API or a library like socket.io-client is more appropriate.51
* **Backend (Python with FastAPI):**
  + **Framework Choice:** FastAPI is the recommended framework for the backend over alternatives like Flask. Its primary advantages are its native support for asynchronous programming (async/await) and its first-class integration with WebSockets, making it exceptionally well-suited for handling the concurrent, I/O-bound operations of a real-time streaming application.53
  + **WebSocket Endpoint:** The FastAPI application will define a WebSocket endpoint (e.g., /ws/recite). When a client connects, this endpoint will handle the persistent connection, receiving audio chunks and sending back analysis results.
  + **Riva Integration:** The backend will function as a gRPC client to the locally running Riva server. It will manage the streaming request to the Riva ASR service and process the stream of responses.45
  + **Business Logic:** The backend is the orchestrator of the entire inference pipeline described in section 4.1. It receives audio from the client via WebSocket, forwards it to Riva via gRPC, receives transcriptions and timestamps, dispatches the audio segments to the custom Tajweed models for analysis, and streams the final, structured feedback back to the client via the WebSocket.
* **Database (MongoDB):**
  + **Schema Design:** A NoSQL document database like MongoDB is an excellent choice for storing the results of the recitation analysis. Its flexible, JSON-like document model can easily accommodate the complex, nested, and sometimes variable structure of the analysis data.57
  + **Data Collection:** A primary collection, which could be named recitation\_analyses, will store the results. Each document in this collection can represent a single analyzed word from a user's recitation session. This granular storage allows for powerful analytics and personalized feedback over time. The schema should be designed with querying and time-series analysis in mind, using fields like user\_id and created\_at as primary indexes.59

**Table 4: Proposed MongoDB Schema for Recitation Analysis Data**

This table provides a concrete, well-structured schema for storing the application's output data. This is crucial for providing user history, generating analytics, and creating datasets for future model retraining. The proposed schema is designed to be both human-readable and query-efficient, using nested arrays to naturally represent the relationship between a word and its associated Tajweed rules.

JSON

{  
 "\_id": ObjectId("..."),  
 "session\_id": ObjectId("..."), // Foreign key to a parent recitation session document  
 "user\_id": ObjectId("..."), // Foreign key to the user document  
 "ayah\_key": "2:255", // String identifier for the verse (Surah:Ayah)  
 "word\_index\_in\_ayah": 5, // The 0-indexed position of the word in the ayah  
 "word\_text\_uthmani": "الْحَيُّ", // The word as it appears in the Uthmani script  
 "timestamps": {  
 "start\_ms": 1250, // Start time of the word in the audio stream (milliseconds)  
 "end\_ms": 1600 // End time of the word in the audio stream (milliseconds)  
 },  
 "audio\_segment\_ref": "s3://bucket/path/to/segment.wav", // Reference to the extracted audio chunk  
 "annotations":,  
 "created\_at": ISODate("...") // Timestamp for the analysis record  
}

### **4.3 Future Work: Expanding Beyond Madd and Qalqalah**

The architecture and methodologies established for detecting Madd and Qalqalah create a robust foundation for significant future expansion. The long-term vision for TajweedAI should involve progressively increasing its rule coverage and the sophistication of its feedback.

* **Extending Rule Coverage:** The hybrid modeling approach is extensible. Other Tajweed rules that have distinct acoustic signatures can be targeted next. For example, rules involving nasalization, such as *Ghunnah* on a doubled Meem or Noon, or the various forms of *Ikhfaa* (hiding the Noon sound), produce unique spectral patterns. These could be detected using a similar approach to Qalqalah: extracting MFCCs and training a dedicated CNN classifier to recognize their specific phonetic signatures.
* **Developing More Nuanced Feedback:** The initial system provides a binary "Correct/Incorrect" classification. A future iteration could move towards more granular, quantitative feedback. For instance, instead of just "Incorrect (Too Short)," the system could provide a regression-based output like "Madd duration was 1.5 harakat (expected 2)." This would offer learners more precise guidance for correction.
* **Personalized Learning and Analytics:** The MongoDB schema is designed to capture rich data on user performance. By aggregating this data over time, TajweedAI can develop personalized learning paths. It could identify a user's most common mistakes (e.g., "consistently recites Madd Muttasil for only 3 harakat") and recommend specific verses or rules to practice, mirroring the data-driven guidance that Tarteel provides for memorization.5
* **Model Refinement through a Feedback Loop:** With user consent, recordings flagged as incorrect could be collected to create a large, diverse dataset of genuine recitation errors. This dataset would be invaluable for retraining and improving the custom Tajweed detectors, creating a virtuous cycle where the system becomes more accurate as more people use it. This mirrors the expert-in-the-loop refinement process described in academic literature, but on a crowd-sourced scale.41

## **Conclusion**

This technical roadmap has laid out a comprehensive and feasible blueprint for the development of TajweedAI, a specialized system for Quranic recitation mistake detection. By grounding the project in a rigorous acoustic-phonetic framework and leveraging a powerful, synergistic ecosystem of tools and data, the path from concept to a functional prototype is rendered clear and achievable for an individual developer.

The analysis concludes that the project's feasibility hinges on several key strategic decisions. First, the adoption of the NVIDIA NeMo and Riva stack provides access to state-of-the-art speech AI capabilities that can be run locally. The discovery that NVIDIA's flagship pre-trained Arabic ASR model was trained on Tarteel's own dataset is a critical enabler, dramatically reducing the complexity and resource requirements of the project. Second, the rich data provided by Tarteel's Quranic Universal Library (QUL), when combined with publicly available textual annotations like those from the cpfair/quran-tajweed repository, makes the construction of a high-quality, ground-truth dataset possible through an automated data fusion pipeline. Finally, the proposed hybrid modeling strategy—using a classical SVM for the one-dimensional problem of Madd duration and a lightweight CNN for the multi-dimensional patterns of Qalqalah—is a pragmatic engineering choice that balances performance with the computational constraints of a personal computer.

The proposed long-term architecture, featuring a React frontend, a FastAPI backend with WebSocket communication, and a flexible MongoDB database, provides a scalable and robust foundation for a real-time, interactive user experience. By successfully implementing this roadmap, TajweedAI can move beyond the current paradigm of word-level error detection and offer the kind of granular, phonetic-level feedback that learners need to truly perfect their recitation. This project not only represents a significant technical challenge but also holds the potential to create a deeply impactful tool that can aid Muslims worldwide in their sacred relationship with the Holy Quran.

#### Works cited

1. Automated Detection and Consistency Analysis of Tajweed Recitation Rules in the Holy Quran (2022) | Karim Aly Mohammad - SciSpace, accessed June 20, 2025, <https://scispace.com/papers/automated-detection-and-consistency-analysis-of-tajweed-1zv3t1b0>
2. Quran Memorization Journey Tips — Tajweed Reflection and Correct Recitation - Tarteel, accessed June 20, 2025, <https://tarteel.ai/blog/tajweed-and-reflection-for-hifz/>
3. Automated tajweed checking rules engine for Quranic learning - SciSpace, accessed June 20, 2025, <https://scispace.com/pdf/automated-tajweed-checking-rules-engine-for-quranic-learning-2a3g3h08gf.pdf>
4. The Role of the Tarteel Application in Maintaining the Memorization of Al - ejournal unira malang, accessed June 20, 2025, <https://ejournal.uniramalang.ac.id/alwijdan/article/download/5640/3403/34148>
5. How to Memorize the Quran by Yourself: Complete Guide to Self-Guided Hifz, accessed June 20, 2025, <https://tarteel.ai/blog/how-to-memorize-the-quran-by-yourself-2/>
6. AI & Hifdh - Reddit, accessed June 20, 2025, <https://www.reddit.com/r/Hifdh/comments/1igfopa/ai_hifdh/>
7. Speech AI Solutions and Applications - NVIDIA, accessed June 20, 2025, <https://www.nvidia.com/en-eu/ai-data-science/solutions/speech-ai/>
8. Introducing NVIDIA Riva: A GPU-Accelerated SDK for Developing Speech AI Applications, accessed June 20, 2025, <https://resources.nvidia.com/en-us-riva/intro-blog>
9. Quranic Universal Library - Tarteel, accessed June 20, 2025, <https://qul.tarteel.ai/>
10. Introducing QUL: Your Go-To Hub for Digital Quranic Resources! - Tarteel, accessed June 20, 2025, <https://tarteel.ai/blog/qul-launch/>
11. Exploring the Basics: Types of Madd in Tajweed | Noor Institute, accessed June 20, 2025, <https://www.noorinstitutes.com/types-of-madd-in-tajweed/>
12. Types of Madd - Al-Walid Academy, accessed June 20, 2025, <https://www.alwalidacademy.com/types-of-madd>
13. Understanding Madd e Asli: The Foundation of Quranic Recitation - Studio Arabiya, accessed June 20, 2025, <https://studioarabiya.com/understanding-madd-e-asli/>
14. Quran Tajweed Rules - Top 14 Ahkam With Examples - Shaykhi Academy, accessed June 20, 2025, <https://shaykhi.com/blog/9-basic-tajweed-rules-for-beginners-simplified/>
15. What is a harakat and/or count? - Islam Stack Exchange, accessed June 20, 2025, <https://islam.stackexchange.com/questions/16911/what-is-a-harakat-and-or-count>
16. How long should it be? #learntajweed #learnquran #arabic101 #tajweed - YouTube, accessed June 20, 2025, <https://www.youtube.com/watch?v=anEbtfb2lSM>
17. Consistency Analysis of Madd rules in the Holy Quran | Business Administration School, accessed June 20, 2025, <https://ba.nu.edu.eg/publications/consistency-analysis-madd-rules-holy-quran>
18. Understanding The Tajweed and Its Rules - Equranekareem, accessed June 20, 2025, <https://equranekareem.com/understanding-the-tajweed-and-its-rules>
19. Al-Qalqalah - Tajweed Me, accessed June 20, 2025, <https://tajweed.me/2011/01/04/qalqalah/>
20. Unlocking Al-Qalqala: The Art of Resonance | Free Tajweed Rules Course - Al-dirassa - Best Online Arabic,Islamic Programs and Holy Quran, accessed June 20, 2025, <https://al-dirassa.com/en/al-qalqala-the-resonance-tajweed-rules/>
21. (PDF) The acoustics of Qalqalah in the recitation of the Holy Quran, accessed June 20, 2025, <https://www.researchgate.net/publication/385880379_The_acoustics_of_Qalqalah_in_the_recitation_of_the_Holy_Quran>
22. Automatic Detection of Some Tajweed Rules - ResearchGate, accessed June 20, 2025, <https://www.researchgate.net/publication/369954029_Automatic_Detection_of_Some_Tajweed_Rules>
23. Mispronunciation Detection of Basic Quranic Recitation Rules using Deep Learning - arXiv, accessed June 20, 2025, <https://arxiv.org/abs/2305.06429>
24. Tarteel: Your A.I. Companion for Quran Study | Deepgram, accessed June 20, 2025, <https://deepgram.com/ai-apps/tarteel>
25. Quran script - Quranic Universal Library, accessed June 20, 2025, <https://qul.tarteel.ai/resources/quran-script>
26. About QUL Tools - Quranic Universal Library, accessed June 20, 2025, <https://qul.tarteel.ai/tools>
27. TarteelAI/quranic-universal-library: A comprehensive collection of Quran resources - GitHub, accessed June 20, 2025, <https://github.com/TarteelAI/quranic-universal-library>
28. Tajweed tool - Quranic Universal Library, accessed June 20, 2025, <https://qul.tarteel.ai/tajweed_words/92:5:3>
29. Tajweed tool - Quranic Universal Library, accessed June 20, 2025, <https://qul.tarteel.ai/tajweed_words/2:88:3>
30. nvidia/stt\_ar\_fastconformer\_hybrid\_large\_pcd\_v1.0 · Hugging Face, accessed June 20, 2025, <https://huggingface.co/nvidia/stt_ar_fastconformer_hybrid_large_pcd_v1.0>
31. cpfair/quran-tajweed: Tajweed annotation for the Qur'an - GitHub, accessed June 20, 2025, <https://github.com/cpfair/quran-tajweed>
32. What is “Hafs”? – Tarteel AI Quran Memorization, accessed June 20, 2025, <https://www.tarteel.ai/glossary/hafs>
33. Pause marks and sajdah signs · Issue #2 · cpfair/quran-tajweed - GitHub, accessed June 20, 2025, <https://github.com/cpfair/quran-tajweed/issues/2>
34. NeMo Forced Aligner (NFA) — NVIDIA NeMo Framework User Guide, accessed June 20, 2025, <https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/tools/nemo_forced_aligner.html>
35. NeMo Forced Aligner and its application to word alignment for subtitle generation - ISCA Archive, accessed June 20, 2025, <https://www.isca-archive.org/interspeech_2023/rastorgueva23_interspeech.pdf>
36. Comparing Deep Learning and Traditional Machine Learning - The CEO Views, accessed June 20, 2025, <https://theceoviews.com/comparing-deep-learning-and-traditional-machine-learning/>
37. Comparison of machine learning and deep learning for view identification from cardiac magnetic resonance images - PubMed, accessed June 20, 2025, <https://pubmed.ncbi.nlm.nih.gov/34813989/>
38. Praat short tutorial by - Stanford University, accessed June 20, 2025, <https://web.stanford.edu/dept/linguistics/corpora/material/PRAAT_workshop_manual_v421.pdf>
39. Automatic measurement of vowel duration via structured prediction - PMC, accessed June 20, 2025, <https://pmc.ncbi.nlm.nih.gov/articles/PMC5392101/>
40. 275 MFCC-VQ APPROACHFOR QALQALAH TAJWEED RULE CHECKING Ahsiah Ismail 1, Mohd Yamani Idna Idris 2, Noorzaily Mohamed Noor 3, Zaid - The International Journal of East Asian Studies, accessed June 20, 2025, <https://ijeas.um.edu.my/index.php/MJCS/article/download/6829/4501>
41. smartajweed automatic recognition of arabic quranic ... - arXiv, accessed June 20, 2025, <https://arxiv.org/pdf/2101.04200>
42. NVIDIA Announces Riva Speech AI and Large Language Modeling Software For Enterprise, accessed June 20, 2025, <https://developer.nvidia.com/blog/nvidia-announces-riva-speech-ai-and-large-language-modeling-software-for-enterprise/>
43. Riva Skills Quick Start - NVIDIA NGC, accessed June 20, 2025, <https://catalog.ngc.nvidia.com/orgs/nvidia/teams/riva/resources/riva_quickstart>
44. Getting Started — NVIDIA NIM Riva NMT, accessed June 20, 2025, <https://docs.nvidia.com/nim/riva/nmt/latest/getting-started.html>
45. asr-basics.ipynb - nvidia-riva/tutorials - GitHub, accessed June 20, 2025, <https://github.com/nvidia-riva/tutorials/blob/main/asr-basics.ipynb>
46. Automatic Speech Recognition (ASR) — NVIDIA NeMo Framework User Guide, accessed June 20, 2025, <https://docs.nvidia.com/nemo-framework/user-guide/24.09/nemotoolkit/asr/intro.html>
47. Build Real-Time Audio Streaming with React and Backend Integration - AnyLearn.ai, accessed June 20, 2025, <https://anylearn.ai/guide/21ecb789-4340-4ecb-a726-94c2f7e5bc3e>
48. How to implement audio streaming in React + Node using the OpenAI TTS model, accessed June 20, 2025, <https://stackoverflow.com/questions/79294027/how-to-implement-audio-streaming-in-react-node-using-the-openai-tts-model>
49. How to Build a Speech-to-Text App with React and Transformers.js ..., accessed June 20, 2025, <https://dev.to/emojiiii/how-to-build-a-speech-to-text-app-with-react-and-transformersjs-4n1f>
50. Picovoice/react-native-voice-processor: React Native audio recording package designed for real-time speech audio processing - GitHub, accessed June 20, 2025, <https://github.com/Picovoice/react-native-voice-processor>
51. react-speech-recognition - npm, accessed June 20, 2025, <https://www.npmjs.com/package/react-speech-recognition>
52. How to Add Speech Recognition to Your React and Node.js project - Deepgram Blog ⚡️, accessed June 20, 2025, <https://deepgram.com/learn/how-to-add-speech-recognition-to-your-react-project>
53. NLP Server — ACE Agent - NVIDIA Docs, accessed June 20, 2025, <https://docs.nvidia.com/ace/ace-agent/4.1/user/nlp-server.html>
54. Python FastAPI Tutorial: Build a REST API in 15 Minutes - YouTube, accessed June 20, 2025, <https://www.youtube.com/watch?v=iWS9ogMPOI0>
55. How to Deploy an AI Model in Python with PyTriton | NVIDIA Technical Blog, accessed June 20, 2025, <https://developer.nvidia.com/blog/how-to-deploy-an-ai-model-in-python-with-pytriton/>
56. nvidia-riva/python-clients - GitHub, accessed June 20, 2025, <https://github.com/nvidia-riva/python-clients>
57. Schema Validation - Database Manual - MongoDB Docs, accessed June 20, 2025, <https://www.mongodb.com/docs/manual/core/schema-validation/>
58. Building AI with MongoDB: Giving Your Apps a Voice | MongoDB, accessed June 20, 2025, <https://www.mongodb.com/blog/post/building-ai-mongodb-giving-your-apps-voice>
59. MongoDB Time Series Data, accessed June 20, 2025, <https://www.mongodb.com/resources/products/capabilities/mongodb-time-series-data>
60. Time Series - Database Manual - MongoDB Docs, accessed June 20, 2025, <https://www.mongodb.com/docs/manual/core/timeseries-collections/>