ChatGPT

Tajweed AI Web Application for Quranic Recitation Feedback

Abstract

This paper presents a comprehensive plan for **Tajweed AI**, a web application that detects and provides feedback on common Tajweed mistakes in Quranic recitation. Focusing on beginner-level Tajweed rules – including Noon Sakinah & Tanween (Idhār, Idghām, Iqlāb, Ikhfā), Noon & Meem Mushaddad (Ghunnah), Meem Sakinah rules (Ikhfā Shafawī, Idghām Shafawī, Izhar Shafawī), **Qalqalah**, and basic **Madd** (2, 4, 6 beat prolongations) – the system leverages recent advances in speech recognition and natural language processing. We survey publicly available datasets such as Tarteel's Quranic Universal Library (QUL) and community-contributed audio datasets (e.g. QDAT and Kaggle Tajweed corpora) to inform data collection and annotation. We explore **ASR** and **NLP** model architectures, comparing **NVIDIA Riva/NeMo** platforms with open-source alternatives like OpenAI's Whisper and HuggingFace wav2vec2 models. Our methodology covers data preprocessing, supervised learning for Tajweed rule classification, and real-time (or near-real-time) feedback generation. Key performance metrics – including Word Error Rate (WER) for speech transcription accuracy, Tajweed mistake detection precision/recall, and system latency – are defined to align with the needs of an interactive tutoring tool. We also outline UI/UX requirements where users record their recitation, the system analyzes pauses, and targeted feedback is given; real-time on-screen transcription is considered for future versions. Ethical considerations such as respectful handling of Quranic audio, user privacy, and the assistant role of AI in religious learning are discussed. We provide an overview of **Tarteel's tech stack** and suggest integration paths, as the ultimate goal is to develop a compelling prototype by the end of the summer (June–September) suitable for Tarteel's QUL grant program and to demonstrate capabilities for potential employment at Tarteel.

Background

Correct Quranic recitation relies on applying **Tajweed rules**, a codified set of pronunciation guidelines that ensure the Quran is recited as it was revealed [1]. Traditionally, mastering Tajweed requires a qualified teacher to listen and correct mistakes, but with millions of learners worldwide, access to expert teachers is limited [1]. Key beginner Tajweed rules involve: - **Noon Sakinah & Tanween**: e.g. Idhār (clear pronunciation of ن/تنوين before throat letters), Idghām (merging the ن/تنوين into certain following letters with/without nasalization), Iqlāb (converting ن/تنوين to a hidden م sound before ب), and Ikhfā (hiding the ن sound with nasalization before other letters). - **Noon & Meem Mushaddad**: pronouncing نّ or مّ (noon/mīm with shaddah) with ghunnah (nasal sound) for two beats. - **Meem Sakinah** rules: Ikhfā' Shafawī (hiding meem at labial point before **ب** with ghunnah), Idghām Shafawī (merging consecutive **م** with ghunnah), and Ith-hār Shafawī (clear meem when followed by other letters). - **Qalqalah**: "bouncing" echo sound on ق ط ب ج د when they carry sukoon (usually at stop). - **Madd (Prolongation)**: extending vowel sounds by the correct length (e.g. 2 beats for normal madd, 4 beats for mandatory elongation with hamzah, and 6 beats for elongation with a sukoon, etc.).

Beginners often struggle with these rules, for example failing to nasalize when required or not elongating madd for the proper duration. To assist learners, prior research has attempted automatic detection of Tajweed rule violations. However, earlier works faced limitations: they often used private datasets and did not leverage sequence modeling for the time-dependent nature of speech [2]. Recent studies have started addressing these gaps by using public datasets and deep learning (e.g. LSTM-based models) to detect mispronunciations of certain rules with high accuracy [3]. Still, many Tajweed rules remain under-studied in automated feedback systems.

Tarteel, an AI-powered Quran app, demonstrated the feasibility of using speech recognition to aid memorization and recitation [4]. Tarteel's current system can follow along as users recite and highlight word-level mistakes (such as missing or incorrect words) in real time [5] [6]. However, it does *not yet support letter-level or Tajweed mistake detection* [7], meaning nuances like pronunciation and application of rules are not flagged. This project aims to fill that gap by focusing on Tajweed-specific feedback. The need for accuracy is paramount – Quranic recitation carries spiritual significance, and any AI feedback must be reliable to avoid reinforcing incorrect habits [8]. By building on advances in Arabic speech recognition and leveraging curated Quranic datasets, we propose a system that can provide beginner reciters with immediate, targeted feedback on Tajweed errors, complementing traditional learning and potentially scaling access to quality Quran education.

Methodology

We propose a hybrid **Automatic Speech Recognition (ASR)** and **Tajweed classification** approach. At a high level, the system will transcribe a user's recitation and simultaneously analyze the audio for compliance with Tajweed rules. The methodology consists of several components and phases:

- **Data Collection & Annotation:** We will aggregate Quranic recitation audio data from public sources. Using Tarteel's **Quranic Universal Library (QUL)** as a foundation, we obtain authentic recitations with word-by-word timestamps [9]. In addition, dedicated Tajweed-focused datasets (such as the QDAT dataset and a Kaggle Tajweed dataset) provide labeled examples of correct vs. incorrect rule pronunciation [10]. We will preprocess this audio (resampling, normalizing volume) and create annotations for Tajweed rules. For example, if a verse contains a noon-sakinah followed by a letter that triggers Ikhfā, we label the expected rule ("Ikhfā") at that timestamp; if the dataset provides an incorrect recitation, we label the type of mistake (e.g. "Idhār applied instead of Ikhfā"). Community contributions (crowdsourced recitations, possibly from non-native speakers) will enrich the variety of mistakes in the training data.

- **Feature Extraction:** From each recorded recitation, we may extract acoustic features that are informative for Tajweed detection. Traditional features like Mel-Frequency Cepstral Coefficients (MFCCs) have been successfully used to detect pronunciation mistakes in Quranic recitation [10]. These features capture the spectral characteristics of audio and can highlight, for instance, the presence or absence of nasal resonance (for ghunnah) or the abrupt stop vs. continued vowel for madd. We will compute MFCC or mel-spectrogram features frame-by-frame as input to learning models. Additionally, forced alignment or dynamic time warping against the Quran's known text can help pinpoint the audio segments corresponding to each letter, which aids in precisely evaluating rule application.

- **Model Training:** Our system will employ a two-stage modeling approach. First, an **ASR model** will be used to convert the speech to text and align it with the expected Quranic verse. This ensures we know which part of the recitation corresponds to which letter/word, providing context for which Tajweed rules should apply. Second, a **Tajweed rule classifier** will assess whether each applicable rule was followed. We envision training supervised classifiers for each targeted rule (or a single multi-class model that outputs the status of all rules). For instance, an LSTM-based classifier could take the audio frames around a noon sound and predict "correct" or "incorrect" application of Ikhfā [10]. We will explore both classical ML (SVM, random forest) and deep learning (CNN/LSTM) for this task, taking inspiration from recent studies that achieved ~95% accuracy on similar tasks using LSTM models [11]. The training process will involve balancing the dataset (since some rules might have more examples than others) and possibly augmenting data (e.g., artificially altering audio to simulate different mistake types) to improve robustness.

- **Real-Time Inference & Feedback:** The ultimate application demands low-latency processing so that users receive feedback shortly after reciting. We aim for near-real-time analysis, ideally under a couple of seconds for a short verse. The ASR and Tajweed models will be optimized for fast inference (using techniques like quantization or running on GPU). During inference, when a user finishes reciting or pauses, the system will finalize the transcript and run the Tajweed checks. Detected mistakes will be mapped to user-friendly feedback, e.g. "Apply Ikhfā: the letter نْ at **…** should be hidden with a nasal sound." The methodology ensures the feedback is specific (pointing out the rule and location) and constructive (possibly suggesting how to correct it).

- **Evaluation & Iteration:** Throughout development, we will evaluate the ASR component on transcription accuracy (especially on Quranic Arabic, which may have unusual proper nouns and elongations), and evaluate the Tajweed detection on labeled data. Mistake detection performance will be measured in terms of precision (are the flagged mistakes actually mistakes?) and recall (does the system catch most of the mistakes?). If performance is lacking for certain rules, we may refine the methodology by collecting more data for those rules or tuning model architectures (e.g., using a more sensitive acoustic model for nasalization detection). The modular design allows iterative improvement of each component (ASR or classifier) independently.

Architecture

To implement the above methodology, we propose a modular system architecture, illustrated in **Figure 1**. The architecture integrates speech recognition with a Tajweed analysis pipeline:

Figure 1: System architecture pipeline for the Tajweed AI application. The user's recitation is first transcribed by an ASR engine, then analyzed for Tajweed rule compliance, and finally the system returns targeted feedback to the user.

As shown, the **Audio Input** from the user (recorded recitation) flows into the **ASR Engine (Speech-to-Text)**. This component can be powered by a state-of-the-art model such as a Conformer neural network or transformer-based ASR trained on Arabic Quranic recitations. Its job is to produce a transcript of the recited verse and, importantly, to align each spoken word (and phoneme) with the timeline of the audio. The transcript can be used to verify if the user recited the correct verse and to locate the positions of letters where Tajweed rules apply (for example, identifying all occurrences of noon sākinah or madd in the verse).

Next, the audio (and alignment info) is passed to the **Tajweed Rule Analyzer**. This module contains classifiers or rule-specific detectors. It examines segments of the audio corresponding to where a rule should be applied: - For a rule like Ikhfā, the analyzer will focus on the portion of audio where a noon sākinah + following letter occur, and determine if the noon was nasalized (correct Ikhfā) or pronounced clearly (mistake). - For Qalqalah, it will analyze the end of the word with a quiescent Qalqalah letter to see if a bouncing sound is present in the waveform. - For Madd, it can measure the duration of vowel sounds and compare against expected length (2, 4, 6 beats, where a beat corresponds to a certain length in milliseconds depending on recitation speed).

The Tajweed Analyzer may employ a combination of signal processing heuristics and machine learning. For instance, it might use a neural network that directly classifies audio segments as correct/incorrect, or simpler threshold-based logic for well-defined features (like madd duration). The architecture is designed to be extensible: new rules (intermediate or advanced) can be added by introducing new detectors without altering the core pipeline.

Finally, based on the analyzer's outputs, the system generates **Feedback to the User**. The feedback component translates detection results into meaningful messages and UI elements. It could highlight specific words or letters on the Quran text displayed to the user and attach a note about the mistake (e.g., coloring a letter red and showing a tooltip explanation). The feedback can also be spoken or written guidance on how to correct the mistake. Since the analysis happens quickly, the user can receive this feedback immediately after recitation, enabling a reflective learning loop (they can try reciting again correctly).

From a deployment perspective, this architecture can be implemented with a backend server that handles the heavy ASR and analysis computation, communicating with a front-end web or mobile app. Given the limited hardware for our prototype, we will likely run the server on a machine with an NVIDIA GPU to leverage accelerated inference, and have the web front-end make requests (audio upload) and receive results. The design also considers scalability: if the system needs to handle many users, each of the components (ASR, analyzer) could be scaled horizontally or optimized (e.g., using batching or streaming ASR for longer recitations).

In summary, the architecture cleanly separates concerns: speech-to-text conversion and Tajweed rule evaluation. This separation allows us to integrate best-in-class ASR models and focus our custom development on the Tajweed-specific logic. It also mirrors the way a human teacher operates – first listening to *what* was recited (words) and then checking *how* it was recited (pronunciation rules).

Dataset Review

A successful Tajweed AI system depends on quality data that covers both correct and incorrect applications of the rules. We survey several datasets and resources:

- **Quranic Universal Library (QUL):** Tarteel's QUL [9] is a treasure trove of Quranic data, including the complete Quranic text, various recitations, and *word-by-word timestamps*. From QUL, we can obtain audio of skilled reciters (which serve as **correct reference** examples of Tajweed application). The timestamped alignments are particularly useful: they allow us to automatically slice audio at exact letter boundaries, which can help generate training samples (e.g., extracting a snippet of audio for a known Ikhfā instance and labeling it as "correct"). However, QUL's recitations are typically error-free; to learn to detect mistakes, we need examples of *incorrect* recitation as well.

- **QDAT Dataset:** The Quranic Audio Dataset (QDAT) is a public dataset introduced in 2021 that specifically targets Tajweed rule mispronunciations [12]. It contains over **1,500 audio samples** of recitation, each labeled for correctness or incorrectness of certain Tajweed rules [13]. Notably, QDAT includes three types of rules: *"Separate stretching"* (a type of madd, likely corresponding to a madd munfasil – a prolongation where the elongation is separated by a word boundary), *"Tight Noon"* (noon mushaddad with ghunnah), and *"Hide"* (Ikhfā) [13]. Each audio either correctly applies the rule or deliberately mispronounces it. For example, an audio labeled "Hide - incorrect" might feature a clear Noon sound where Ikhfā was expected. The creators of QDAT intended it as a benchmark for Tajweed classification, and a recent study achieved about **95-96% accuracy** in detecting those rule violations using an LSTM model [11]. We will use QDAT to train and validate our rule detection models for Ikhfā and Ghunnah rules, and as a template for how to structure our own data for other rules.

- **Kaggle Tajweed Datasets:** Kaggle hosts at least one relevant dataset contributed by the community. One such dataset (by user *alawdisoft*) focuses on **Noon Sakinah and Tanween** rules. It likely contains audio samples categorized by rule (Idhār, Idghām, Ikhfā, Iqlāb) or by correctness of those rules. Although we could not retrieve the full details in this environment, the description suggests it provides audio for each category, which is valuable for a multi-class classification approach (i.e. identifying which rule is applied in a given context). Another Kaggle entry shows efforts to fine-tune ASR models (like Whisper) on Quranic recitations with Tajweed labels, indicating an active community interest. We will explore these Kaggle sources for any labeled data we can incorporate. Even small datasets (a few hundred samples) are useful for transfer learning, especially if they cover rules like Qalqalah or Meem Sakinah which are not in QDAT.

- **Crowdsourced Recitations and Non-Native Speakers:** A 2024 study introduced a **Crowdsourced Quranic Audio Dataset** with annotations from non-Arabic speakers' recitations [14]. This kind of dataset can expose common mistakes made by learners (e.g., mispronouncing certain letters, or applying rules incorrectly due to influence from native language). If accessible, such data will be very valuable for making our system robust to accent and varying recitation quality. In absence of an open dataset, we may conduct a small-scale data collection by inviting volunteers (or using ourselves) to record deliberate mistakes. For example, record one verse multiple times, each time intentionally violating a specific rule, to create a synthetic training sample for that mistake. Care must be taken to ensure the mistakes are realistic (sounding like a typical learner's error).

- **Tarteel Dataset:** Over its development, Tarteel collected a large volume of user recitation data (75,000 minutes of audio as of early 2022) to train its ASR models [5]. While that dataset is not publicly available in full, Tarteel has mentioned highly curated subsets. For our prototype, we will rely on public alternatives as described above, but if collaboration with Tarteel is possible, integrating some of their data (even indirectly via pre-trained models) could significantly boost performance.

Each dataset will undergo **preprocessing**. This includes normalizing audio formats (sample rate, bit depth), trimming silences, and splitting verses if needed. We will also unify the annotation schema: for instance, using a consistent label set for rules and errors (perhaps a JSON structure like `{ start_time: 2.5s, rule: "Ikhfa", correct: false }` attached to an audio). This will allow training models that generalize across datasets. A potential challenge is that different datasets may have different pronunciations (Egyptian vs. Saudi reciters, etc.) and different audio quality. We will mitigate this by filtering out very low-quality audio and possibly using data augmentation (adding noise, small pitch shifts) to make models more robust.

In summary, our data strategy is to combine **expert recitations** (for correct examples and acoustic patterns) with **learner recitations** (for incorrect examples). By covering both ends, the model can learn to distinguish fine-grained differences that signal a Tajweed mistake. The use of public datasets like QDAT also ensures that our results can be benchmarked and the system can be objectively evaluated against prior research.

Model Design

Designing the AI models involves two main parts: the speech recognizer and the Tajweed mistake detector. We consider various architectural options for each, balancing performance with the constraints of a small team and limited compute resources.

1. ASR Model (Speech Recognition): The ASR is the backbone, providing the transcription and alignment. We have several choices: - *NVIDIA Riva ASR (Conformer-CTC):* NVIDIA provides pre-trained Arabic ASR models via Riva, such as a Conformer model trained on thousands of hours of Arabic speech [15] [16]. These models achieve top-tier WER on Arabic benchmarks and output text in Arabic script [16]. Using Riva's API, we can get real-time streaming transcription on an NVIDIA GPU, with reported latencies under 200ms for inference [17]. The advantage of Riva is efficiency and easy deployment (it's optimized for production), plus the ability to fine-tune with NeMo if needed. The drawback is that Riva is proprietary; however, it's free to use with an NVIDIA GPU and aligns with Tarteel's stack which already leverages NVIDIA tech [17]. - *NVIDIA NeMo Toolkit:* NeMo is an open-source toolkit for training and fine-tuning speech models. With NeMo, we could fine-tune a Conformer-CTC model on Quranic recitation data (for example, using QUL's audio or the Tarteel dataset if available). Fine-tuning could improve accuracy on the unique pronunciation and proper nouns of Quranic text. We can also train smaller models like QuartzNet or Citrinet on our data if needed. NeMo integration would allow us to experiment with end-to-end models (like adding a secondary output head for Tajweed classification, though that might be ambitious in a prototype). Given our limited time, we likely will not train from scratch, but NeMo gives the flexibility to adapt existing models. - *Open-Source ASR (Whisper & Others):* OpenAI's Whisper model is a compelling alternative – it's a pre-trained ASR that handles Arabic and has been trained on 680k hours of multilingual data. Whisper's strength is accuracy and noise robustness; it might transcribe Quranic Arabic well (including diacritics if we choose, although diacritics can also be added via post-processing). Using a smaller variant of Whisper (like `tiny` or `base`) could allow near-real-time performance on a decent CPU, which is important given our hardware limits. Initial tests from the community (e.g., on Kaggle) show Whisper can transcribe Quran recitations effectively. The downside is Whisper outputs Latin script by default for Arabic; we'd need to configure it to output Arabic script or convert its output to Arabic text for easier rule mapping. Additionally, Whisper is not specifically trained to "expect" perfect Tajweed, so it might not flag a missing ghunnah as an error – but that's fine, as that's the job of our secondary model. - *Other Hugging Face Models:* There are Arabic ASR models on Hugging Face (e.g., Wav2Vec2 models fine-tuned on Arabic speech). These could be lighter-weight and easier to run on CPU. For example, a Wav2Vec2 model trained on MSA might do an acceptable job on Quranic Arabic, though it may have trouble with verses due to the classical language and extended vowels. Still, as a fallback or comparison, we can test one of these.

Given the time constraints, our likely approach is to use a pre-trained ASR as-is (to avoid long training cycles). We might start with Whisper-small (which can run on CPU/GPU and give good accuracy) for initial development, and if GPU is available, test Riva's Conformer model for speed. We will measure WER on a sample of Quran verses to pick the model that offers the best accuracy/latency trade-off. If needed, we will fine-tune on our domain data (for instance, NeMo allows fine-tuning the Conformer on domain text to handle uncommon proper nouns or elongation markers).

2. Tajweed Mistake Detection Model: This is the core innovation of our system. We are essentially building a specialized classifier that, given an aligned audio segment and context, outputs whether the Tajweed rule was correctly applied. Some design considerations: - *Per-Rule Classifiers vs. Unified Model:* We could train separate models for each rule (e.g., one model detects Ikhfā mistakes, another detects Qalqalah mistakes). This could simplify the learning task since each model focuses on one acoustic phenomenon. Indeed, previous research often tackled one rule or a small set at a time [12]. The downside is maintaining multiple models and needing enough data for each. Alternatively, we could design a single multi-output model that looks at a recitation and predicts errors in any of the target rules. This would be complex, as it requires the model to implicitly learn all rules and their contexts. A compromise is a pipeline: first identify where rules *should* apply (using text), then apply a focused model to that segment. We will likely adopt this pipeline approach. - *Model Type:* Considering the time-series nature of audio, **Recurrent Neural Networks (RNNs)** like LSTM or modern variants like Bi-LSTMs have been used effectively [10]. An LSTM can learn the temporal patterns of correct vs incorrect pronunciation (for example, the sustained low-frequency energy of a ghunnah, or the abrupt cutoff that indicates Qalqalah). We can feed it MFCC feature sequences of a fixed window around the event (e.g., 0.5 seconds before and after the letter). Another option is a **Convolutional Neural Network (CNN)** which can learn from spectrogram "images" – this might detect subtle differences in frequency domain (nasalization adds certain formant patterns which a CNN might pick up on a spectrogram). A 2D CNN on a spectrogram is effectively treating it like an image classification (correct vs incorrect). - *Hybrid with ASR:* We will explore if the ASR's acoustic model can be extended to predict Tajweed correctness. For instance, a transformer encoder that is pretrained on speech could be fine-tuned with an extra classification head for rule application. This would resemble a **multi-task learning** setup. Given our resource limits, this might be too ambitious now, but we note it as a future direction (as it could potentially catch errors even without knowing the exact text, by purely acoustic cues). - *Heuristic Features:* In some cases, simple features might work. For Madd, measuring duration (using the alignments) and comparing with an expected range could flag too-short prolongations. For Qalqalah, checking the waveform for a characteristic oscillation at the end could be done via digital signal processing. We will combine these where applicable with the ML approach (a technique known as feature fusion or using heuristic outputs as additional inputs to the ML model). - *Training Strategy:* We plan to train the Tajweed models on a combination of real and synthetic data. Real data from QDAT and Kaggle provides ground truth. To cover rules not in those sets (like Meem Sakinah rules or Idghām), we might generate examples: e.g., take a correct recitation audio and digitally superimpose a clear "n" sound to simulate Idhār where Ikhfā should be, creating an incorrect sample. Or record one of us reading incorrectly on purpose. These augmented samples, while not perfect, can increase the diversity of training data. We must ensure the models don't overfit to specific voices; using mel-scaled features and perhaps adding slight noise can help generalize.

3. Integration and Alternates: The ASR and Tajweed models must work in tandem. One integration approach is to have the ASR produce a lattice or confidence measure – for instance, if the ASR is uncertain whether the user said a noon or not in a spot where a noon should be hidden, that could indicate an Ikhfā issue. If accessible, such low-level info from the ASR (like acoustic probabilities for phonemes) could directly feed into Tajweed detection, improving accuracy. We will check if Riva/Whisper allow extracting intermediate representations.

Another model to consider for alignment and phoneme-level analysis is a **phoneme recognizer** or **forced aligner** (like Montreal Forced Aligner or wav2vec-U). This could give precise timing of each phoneme spoken by the user, which is extremely useful for pinpointing rule application. Due to time, we may stick with alignment via the ASR's recognized text, but using a specialized aligner is an option if timing issues arise.

Finally, all models will be evaluated on a validation set of recitations with known mistakes (if available). We will utilize cross-validation if data is scarce, and monitor for overfitting. The target is not just raw accuracy, but ensuring the model's mistakes, if any, are not in critical areas. For example, a false positive (saying the user made a Tajweed mistake when they did not) could confuse or discourage a user. We'd prefer the model to possibly miss a subtle mistake (false negative) rather than wrongly accuse a correct recitation. We will tune classification thresholds with this in mind, possibly favoring high precision.

In summary, the model design brings together **ASR for content** and **specialized classifiers for form**. By leveraging proven architectures like Conformer ASR and LSTM classifiers – and tailoring them to the Quranic domain with fine-tuning and custom data – we aim to achieve a reliable detection of beginner Tajweed mistakes. Our design choices are guided by practicality (use existing models where possible) and modularity, so each component can be improved or replaced independently as better techniques become available.

Evaluation Metrics

To ensure the Tajweed AI system meets its goals, we define several metrics and evaluation criteria, focusing on both the accuracy of detection and the responsiveness of the system. Key metrics include:

- **Word Error Rate (WER) for ASR:** WER measures the percentage of words incorrectly transcribed by the ASR (substitutions, deletions, insertions). A low WER is crucial because the Tajweed analysis relies on correct alignment of the recitation. If the ASR mis-recognizes words, the system might check the wrong letters for rules or even miss entire opportunities to apply a rule. We will evaluate WER on a test set of Quranic verses (possibly using standard Quranic reciter audio as ground truth transcripts). Our goal is to achieve a WER as low as possible, ideally under 10%. Given that Tarteel achieved highly accurate Quranic ASR by training on 75k minutes of data [5], using their model or similarly trained models should give us a strong baseline. If our chosen ASR's WER is high on certain letters (e.g., confusion between similar-sounding letters), we may note that as a limitation or try to mitigate it via post-processing (for example, knowing that a certain verse must contain a specific word, we could correct obvious errors using text matching).

- **Tajweed Error Detection Accuracy:** For each Tajweed rule, we will measure how well the system detects mistakes. This can be broken down into:

- *Precision*: Of all the instances the system marked as incorrect for a given rule, how many were truly incorrect? High precision means few false alarms. This is important for user trust – we don't want to frequently tell users they made a mistake when they haven't.

- *Recall*: Of all the actual mistakes the user made (according to ground truth labels), how many did the system catch? High recall ensures the tool is effective in identifying issues the user needs to know about.
- *F1-Score*: The harmonic mean of precision and recall, giving a single measure of detection quality per rule.

We expect some variation by rule. For example, detecting a missing ghunnah (nasalization) might be easier (hence higher precision/recall) than detecting a subtle timing difference in madd. We will evaluate these metrics on a labeled test set of recitations (like a portion of QDAT or our own collected clips with known mistakes). An ideal outcome is precision and recall above 90% for each rule, which would be on par with recent research results for basic rules [11]. However, we will set realistic benchmarks (perhaps ~80% for recall and precision initially) and improve over time.

- **Latency:** This metric is about user experience – how long the user waits to get feedback. We will measure the time from when a user finishes reciting (or hits stop) to when feedback is displayed. Our aim is to keep this latency low (a few seconds at most). Specifically, if reciting a single verse (~5 seconds of speech), the feedback ideally appears in <2 seconds. Latency will be affected by:
- ASR processing time (which can be real-time or faster-than-real-time depending on model and hardware).
- Tajweed analysis time (which should be relatively small as the audio segments are short and models lightweight).
- Any communication overhead (if using a server, the round-trip time of audio upload and result download).

We will test latency under realistic network conditions and on our available hardware. If using a GPU on a local server, ASR can often process faster than real-time; for example, Tarteel's system operates with <200ms latency thanks to GPU optimizations [17]. In our budget scenario, using Whisper on CPU might be slower (maybe recitation length *2 or* 3). We can mitigate this by using smaller models or by cutting off processing once enough confidence is reached.

- **Throughput and Scalability:** While not a focus for the prototype, if we demonstrate this to stakeholders (like Tarteel's grant committee), they may ask how it scales. We should measure how many recitations can be processed per minute or hour with our setup. For now, a single-user latency focus is fine, but we can note if the system can handle back-to-back requests smoothly. Our design can batch some operations (though streaming nature of ASR limits that). We will mostly ensure that the system can handle at least the typical use case of one recitation at a time.

- **User-Centric Metrics:** Beyond technical metrics, we consider **usability metrics**: is the feedback understandable and helpful? In a full study, we would do user testing to gather qualitative feedback. For the prototype, we can simulate a user session and see if the highlighting and messages intuitively guide correction. Additionally, if possible, we could measure how many attempts it takes for a user to correct an error after seeing feedback (as a proxy for feedback clarity). These are beyond pure engineering metrics but important for evaluating the educational value of the system.

- **Error Analysis:** As part of evaluation, we will perform error analysis for cases of false positives/negatives. For instance, if the system missed an Ikhfā mistake, was it because the user's pronunciation was very close to correct (i.e., borderline)? Or did noise in the recording cause the model to misjudge? If we find patterns (e.g., always failing on a certain letter or with female voices vs male voices), we'll document those as areas for improvement.

Success criteria for the prototype will be defined by a combination of these metrics. For example: "The system should transcribe verses with WER < 10%, detect at least 80% of intentional Tajweed mistakes in test audio, and have an average feedback latency under 3 seconds." Meeting these will make a strong case in our report to Tarteel's QUL grant committee that the prototype is viable. We will include a small results table in our paper to summarize these metrics once measured (for now we outline what will be measured).

By clearly defining metrics from the start, we ensure that we stay goal-oriented during development – focusing on what will actually make the tool effective for learners (accuracy and speed), rather than just achieving technical milestones.

Implementation Roadmap

Building the Tajweed AI application within a summer (June to September) is ambitious but feasible with careful planning. We outline a **timeline with milestones** for our 2–3 person team, ensuring steady progress and a functional prototype by the end of September. The roadmap is illustrated in **Figure 2** and detailed below:

Figure 2: Project timeline from June to September, with major milestones for each month. The project phases include data preparation, model development, integration, and testing/demonstration.

- **June: Data Collection & Preparation** – In the first month, our focus is on assembling and understanding the data. Tasks:
- Gather datasets: Download QUL resources, QDAT audio, and any relevant Kaggle datasets. Organize them on our storage.
- Annotation setup: Define a unified schema for Tajweed annotations. If needed, create annotation tools or simple scripts to label audio segments (e.g., listening to a subset of clips and marking mistakes to verify dataset labels).
- Data preprocessing: Convert all audio to a common format (e.g., 16 kHz WAV). Perform exploratory analysis – e.g., listen to some correct vs incorrect samples to intuitively grasp differences, plot some spectrograms of mistakes vs correct recitations for insight.
- Define evaluation set: Reserve some data (or create some recordings) for final testing that we will not use in training.

- **Milestone (end of June):** Have a cleaned and annotated dataset ready. Also by this time, finalize the choice of ASR model and have it set up (e.g., able to run a pre-trained ASR to transcribe audio offline).

- **July: Model Prototyping & Selection** – In this month, we develop the core models and choose the best approaches. Tasks:

- Implement or integrate the ASR engine: This might involve setting up NVIDIA NeMo and loading a pre-trained Conformer model, or installing OpenAI Whisper. We will run tests on a variety of sample recitations to gauge accuracy and speed. If adjustments or fine-tuning are needed (and feasible), do them early in the month.

- Develop Tajweed detection models: Start with one rule as a pilot (for example, Ikhfā). Train a classifier (like an LSTM on MFCC features) using QDAT's Ikhfā data. Evaluate its performance. Iterate on the architecture (try a CNN, try adding more context, etc.) until reasonably satisfied. Then extend to other rules one by one. We might prioritize rules that we have data for (Ikhfā, Ghunnah, Madd) and stub the others (for instance, write simple logic for Qalqalah initially).
- Combine pipeline: Write a script that takes an audio file of a recitation, uses ASR to get text, then runs the rule checks, and outputs a structured result (e.g., JSON listing any mistakes). This will be the backend logic for the web app.

- **Milestone (end of July):** Functional prototype of the backend: we can input an audio clip and get a transcript + Tajweed mistake report. We should also have a sense of the model performance (initial metrics) and have identified any major issues. By this point, we aim to have at least the primary rules working (even if accuracy can still be improved).

- **August: Integration & UI/UX Development** – Now we turn the backend into a user-facing application and refine the system. Tasks:

- Develop the web interface: Using a suitable framework (maybe a simple React front-end or even just HTML/JS for prototype), create a page where a user can record or upload recitation audio. Include a text display of the verse (which we can get from QUL by verse reference) and placeholders for feedback (like highlighting or a list of detected mistakes).
- Connect front-end to backend: Set up a lightweight web server (Flask/FastAPI in Python) to host the model. When a user submits audio, the server runs the pipeline and returns results. Ensure to handle audio encoding, file size limits, etc.
- UI feedback mechanisms: Implement the highlighting of mistakes. For example, if a mistake is detected on a particular word or letter, highlight that word in red. Provide a tooltip or a sidebar that says what the mistake was ("Ikhfā not applied: you pronounced the noon clearly. Try hiding it by nasalizing."). Focus on making the feedback clear and motivational.
- Real-time considerations: Though real-time continuous feedback is not required, we might attempt to show interim results, e.g., display recognized words as the user is reciting (since Tarteel does this). If time permits, implement a simple version of this using the ASR partial results.
- Performance tuning: Optimize anything necessary for speed (perhaps loading models into memory once at startup, so each request is faster). If running on CPU is too slow, decide on whether to use a cloud GPU for demos.

- **Milestone (end of August):** End-to-end system demo. A user (or us acting as a user) can open the app, recite a verse, and see the feedback on the screen. All primary features should be in place. At this stage, we should begin preparing for presenting it – gather some example scenarios that show the system working correctly.

- **September: Testing, Evaluation & Demo Prep** – The final month is for polish, evaluation, and preparing our grant submission or presentations. Tasks:

- Thorough testing: Conduct systematic tests with a variety of verses and speakers. Use our reserved test set to compute final metrics (WER, precision/recall for mistakes) and see if our targets are met. Identify any last-minute critical bugs (e.g., app crashes with long audio, or a particular rule always misfires) and fix them.

- User feedback: If possible, have a few people (friends or community members) try the prototype and give feedback on the UX and usefulness of the feedback. This will help us fine-tune the messaging and UI layout.
- Documentation: Write documentation for the code (so that Tarteel engineers can understand how it works, in case of integration). Also, draft the grant proposal or report, which this paper essentially serves as. Ensure that we clearly articulate the value of the prototype and any data on its performance.
- Future plan outline: Given that QUL grantors or Tarteel hiring managers will be interested, we prepare a short roadmap of what we'd do beyond the prototype (e.g., expanding to more rules, building a mobile app, etc.), showing that we have a vision for the project's growth.
- **Milestone (mid-late September):** Deliver the final demonstration to stakeholders. This could be a live demo of the app on a call or an in-person meeting. We'll also provide the written whitepaper (this document) and any supplementary materials. The prototype doesn't need to be perfect, but it should compellingly show that an AI can catch Tajweed mistakes and help learners.

Throughout the project, given our limited team size, we will practice agile development – iteratively building and testing components rather than trying to perfect one big model in isolation. We will also keep an eye on resource usage: using free GPU credits (Google Colab) or a single affordable GPU machine for training small models, and otherwise optimizing code to run inference on CPU if needed.

Risk management: Some anticipated challenges include ASR accuracy on highly melodic recitations (which could confuse the model) and scarcity of training data for certain rules. Our timeline includes buffer in August to adjust – e.g., if a certain model isn't accurate enough by end of July, we allocate time to gather more data or try a different approach in early August. The phased timeline ensures core functionality is ready by end of summer, even if some advanced refinements (like real-time transcription or covering advanced Tajweed) are left for future.

By following this roadmap, we expect to arrive at September with a working Tajweed AI prototype that can impress both the QUL grant committee and serve as a strong portfolio piece for Tarteel, demonstrating our ability to deliver an impactful AI project within a tight timeframe.

UI/UX Design

The user interface and experience are critical for a learning tool like this. We aim to design the UI to be intuitive, educational, and respectful of the Quran's presentation. Here we describe the envisioned UI/UX flow and features:

- **User Recording Flow:** When the user opens the web application, they will be greeted with a simple, focused interface. The Quranic text (either the verse they want to recite or a selection menu to choose a verse) is prominently displayed in a large, legible Arabic font. The user either selects a specific verse or passage to practice, or we provide a default (such as the first verse of a popular surah) as a starting point. A **record button** allows the user to start reciting; this could be accompanied by a prompt like "Press Record and start reciting the verse above". The design here should be minimalistic to avoid distraction – likely just the text, record button, and maybe a skip/next verse button.

- **During Recitation (Optional Real-Time Feedback):** In this prototype, real-time transcription isn't the main focus, but we plan for future support. Possibly, as the user speaks, the recognized words could appear underneath the Arabic text (much like karaoke style). Tarteel's app, for example, highlights words in real-time as it recognizes them [18]. For now, we might simulate this by showing a loading indicator or waveform animation to assure the user that recording is in progress. The act of reciting itself should not need them to stare at the screen – they might be looking at the verse text or have it memorized. We ensure the app can handle pauses or the user stopping early (maybe adding a "Done" button if they want to manually stop).

- **Post-Recitation Analysis:** Once the user finishes (either by hitting stop or a short silence triggers auto-stop), the system will analyze the recitation. The UI should indicate this processing state – e.g., a message "Analyzing your recitation..." with a spinner. This should only last a brief moment (a second or two ideally). Then the results are displayed in an easy-to-digest format. The Quranic text that was recited is shown again, but now any words or letters where a Tajweed mistake was detected are highlighted (e.g., highlighted in red or underlined). For instance, if the user was reciting "وَمَن يَقْنُطُ مِن رَّحْمَةِ رَبِّهِ..." and they failed to do Idghām in the "مِن رَّحْمَةِ" (where a Noon should merge into the Ra), the word containing that noon or the letter itself would be highlighted.

- **Feedback Details:** Alongside the text, a feedback panel or tooltip explains each highlighted segment. The feedback is in simple terms: rather than saying "You made a Tajweed error", it will specify *which rule* and *what should be done*. For example:

- "**Ikhfā**: The letter **ن** in "**مَنْ كَان**" should be hidden (ghunnah) because it's followed by **ك**. Try not to pronounce the 'n' clearly."
- "**Madd**: The elongation on **الضَّالِّين** was too short. It should be held for about 6 beats (longer)."
- "**Qalqalah**: Try to bounce the sound on the **ق** at the end of **الخلق** – it should have a slight echo."

This feedback text can appear when the user clicks on or hovers over a highlight, or it can be listed below the verse as bullet points corresponding to each mistake. We will ensure the tone is encouraging and instructive, not just pointing out error but also guiding correction. If the recitation was perfect or no detectable mistakes, the app can congratulate the user ("Great job! No Tajweed issues detected."), which serves as positive reinforcement.

- **UI Elements and Aesthetics:** We will adhere to respectful design. Quranic text will be displayed with proper orthography and diacritics. We may incorporate a feature to play the correct recitation audio of the verse (maybe a "Hear correct recitation" button using an existing recording) so the user can listen and compare. Colors will be chosen carefully: e.g., green highlights for correct, red for mistakes, or just red for mistakes while leaving correct parts normal. The background and overall theme should be calm and not distracting – likely using earthy or neutral tones common in Quran apps.

- **Responsiveness and Accessibility:** The app should work on desktop and mobile browsers, since many users might want to practice on a phone. We'll use responsive design to ensure the text and buttons resize appropriately. Also, since the app deals with Arabic text and possibly translation for explanations, we'll make sure the fonts are readable and possibly allow toggling a translation of the verse for non-Arabic speakers (though primary users likely know the verse meaning already when memorizing).

- **User Control and Privacy:** The user should feel in control of their data. We will have a clear indicator when recording is active (and reassure that nothing is saved or uploaded beyond analysis, unless they opt in). Also, after feedback is given, the user might want to retry the verse. We could include a "Try Again" button that resets the process, enabling them to implement the feedback immediately. Over time, a log of mistakes could be stored locally (in the browser) so the user can track improvement – but for the prototype, we might keep it session-based.

- **Edge Cases:** If the ASR was very unsure and we can't even tell what verse it was (maybe the user recited something very off or mumbled), the app should handle it gracefully: e.g., "Sorry, I couldn't understand the recitation. Please try again or recite more clearly." Similarly, if multiple mistakes are overlapping, ensure the highlights don't clutter the text (we might need to pick the most significant mistake per word to highlight if there are many).

The UI design takes inspiration from Tarteel's existing app where applicable, but since we are focusing on Tajweed, we extend the design to show letter-level detail. By pausing after recitation to give feedback, we avoid information overload during the act of reciting (which could be distracting). This pause-then-feedback design aligns with how a teacher would first listen fully, then give comments.

In summary, the UX is designed to be a **tutor-like experience**: the app listens patiently, then provides specific, actionable advice. It should feel like a personal tutor highlighting your Quran and jotting notes on it. The respect for the content is maintained by not altering the Quranic text except to overlay non-destructive highlights. If our UI/UX is effective, users should find the tool easy to use and helpful in improving their recitation in an engaging way.

Ethical Considerations

Building an AI for Quranic recitation comes with important ethical and religious considerations. We address these to ensure the project aligns with Islamic etiquette and general data ethics:

- **Accuracy and Responsibility:** As emphasized by Tarteel, when dealing with Quranic recitation, *accuracy is paramount* [8]. A flawed AI that gives incorrect feedback could lead a person to learn something wrong about recitation, which is unacceptable given the Quran's sacredness. Therefore, we have a responsibility to verify the system's outputs. This means rigorously testing the model and, if uncertain about a case, either not giving feedback on it or explicitly saying "unsure" rather than misguiding. In deployment, a disclaimer can be included: e.g., "AI feedback is not a substitute for a qualified teacher. If in doubt, consult a teacher." This sets the expectation that the tool is an aid, not an absolute authority.

- **Role of AI vs Teachers:** We reiterate that the system is meant to **assist, not replace, human instructors**. It serves as a practice tool for students to use independently, especially when a teacher isn't available. The AI can handle repetitive practice and immediate correction, freeing up human teachers to focus on more nuanced instruction. Prior research also views such systems as assistants to teachers, not replacements [19]. We uphold that perspective. In any presentation to religious scholars or community, we will clarify this point to avoid the impression of trying to remove the human element from an age-old learning tradition.

- **Respect for the Quran:** All handling of Quranic content in the app will be done with respect. For instance, if we display the text, we ensure it's accurate and not altered. When highlighting mistakes, we do so in a way that doesn't deface the Quranic text (e.g., we won't cross things out or distort letters – just overlay color that can be removed). Any audio data of Quran we use for training or processing will be treated as sensitive: we won't use it for non-related purposes, and if storing user recitations, we must consider them as containing Quran and treat them with due respect (e.g., ensuring they are not accidentally exposed or misused).

- **Data Privacy:** From a user privacy standpoint, recitations could be personal or sensitive. While a Quran recitation is not private content in the usual sense (it's not personal information like an address), some users may be shy or concerned about how their voice data is used. We should clearly inform users whether their audio is being stored. For the prototype, we likely won't store anything persistently – analysis can be done in-memory and the audio can be discarded immediately after generating feedback. If we do logging for improvement, we will anonymize it (no association with user identity, possibly just keep the audio features or transcriptions). Compliance with data protection norms is important if this becomes a product – we'd need user consent to use their data to improve the AI, etc.

- **Inclusivity and Bias:** We need to ensure the system works for a diverse set of users. This includes variations in gender (male/female voices), age (child reciters may have different pitch), native accent (non-Arabic speakers might pronounce differently). There's a risk the AI could be biased toward the voices it saw in training (e.g., if all training audio were adult male reciters, the system might not perform as well for a 10-year-old girl's recitation). We address this by incorporating diverse data (like the non-native dataset) and testing on different voice samples. Ethically, it's important that AI in religious education be accessible and fair to all users, not just a subset.

- **Transparency:** We should maintain transparency in how the AI works. This is both an ethical and educational point. If the AI flags a mistake, we should be able to explain, in terms of Tajweed rules, why it's flagged. The system's feedback already does this by referencing the rule, which is good. If a user or teacher asks about the AI's decision, we can refer to known Tajweed rules rather than saying "the algorithm thought so" with no reason. This makes the tool's decisions interpretable and grounded in human-understandable rules.

- **Avoiding Overreliance/Misuse:** One ethical consideration is preventing misuse of the system. For example, someone might try to use the AI to "certify" their recitation for an official purpose (like an exam or an ijazah certification). The system is not a certified authority. We must caution that it's a learning aid. Additionally, we should discourage any use of the system's outputs as a way to criticize or shame others' recitation – the tool is for personal improvement, not to be used as a judging device on others. Implementing this practically might mean keeping results private to the user, not sharing scores publicly, etc.

- **Intellectual Property and Permissions:** The Quran text is in the public domain, but certain recitation recordings might be copyrighted by their reciters or publishers. We must ensure that any audio we use, especially in the app (like if we allow "listen to correct recitation" feature), is either from a public domain source or we have permission. QUL likely provides recitations that are licensed for developer use (as it aims to support developers). We will verify licenses of datasets like QDAT or

Kaggle contributions (most Kaggle datasets specify if they're freely usable). Giving proper credit to data sources in our documentation is also an ethical practice in research.

- **Community and Scholarly Approval:** Since Tajweed is a science with established scholars, it would be wise to get feedback from a Tajweed teacher on our approach. This can catch any potential oversight (maybe an edge case rule or a nuance our model might not know). While perhaps not in the prototype phase, eventually an ethical rollout would involve scholarly endorsement or at least review, to ensure the tool aligns with traditional Tajweed teaching. Engaging with the community also builds trust and ensures the AI is augmenting, not clashing with, existing pedagogical methods.

In conclusion, our ethical framework is to treat the Quran and its recitation with **honor and care**, to treat the users with **respect and privacy**, and to position the AI correctly as a helpful tool under human guidance. By planning for accuracy, transparency, and fairness, we aim to create a system that the Muslim community can embrace without hesitation, insha'Allah. If during development we encounter any aspect that might be ethically questionable (for instance, an ML-driven adjustment that isn't easily interpretable), we will weigh its inclusion carefully and opt for the route that maintains trust and respect.

Conclusion

This work has outlined a comprehensive plan to develop a **Tajweed AI web application** – a system that listens to Quranic recitation and provides corrective feedback on fundamental Tajweed rules. By integrating advanced speech recognition technology with domain-specific rule analysis, we aim to bridge the gap between traditional Quran teaching and modern AI assistance. The proposed system focuses on beginner rules (Noon/Meem regulations, Qalqalah, Madd, etc.), providing a strong foundation that can later be expanded to more advanced Tajweed concepts.

We began by surveying the background and need: the shortage of teachers for the ever-growing global Muslim population and the potential of AI to help individuals practice recitation with immediate feedback [1]. Our methodology centers on leveraging publicly available resources – from Tarteel's QUL to academic datasets – ensuring that our approach is built on open and reproducible foundations. We detailed how the **architecture** will function, processing audio in a pipeline that mirrors a human teacher's ears and insights. Through the **dataset review**, we identified key data sources (like QDAT on Kaggle) that will fuel our models, and through **model design**, we charted a path to utilize state-of-the-art ASR (e.g., NVIDIA Conformer via Riva, or Whisper) coupled with tailored classifiers (LSTM/CNN) for rule detection.

Our evaluation plan emphasizes not just raw accuracy but also the real-time performance and user-centric effectiveness of the system. The **UI/UX design** we proposed ensures that the application will be user-friendly and pedagogically sound, giving reciters clear and respectful guidance. We have addressed ethical considerations at every step, acknowledging the responsibility that comes with using AI in the context of the Quran.

Crucially, we have also shown how this project aligns with **Tarteel's tech stack and mission**. Tarteel's existing platform uses deep learning models and NVIDIA optimizations to achieve real-time Quranic speech recognition [5]. Our system can plug into such a stack – for instance, by running our Tajweed analysis after Tarteel's own ASR engine or by deploying our models on the same infrastructure. The modular nature of our design means integration could be as simple as an API call from Tarteel's app to our Tajweed service, feeding back results to be displayed in their interface. This synergy could accelerate Tarteel's roadmap if they plan to introduce letter-level mistake detection in the future, essentially offering them a prototype solution ready to be expanded. As Tarteel's co-founder has indicated enthusiasm for expanding AI's role in Quran study, our project is well-positioned to contribute to that vision.

By the end of the summer, we expect to have a working **prototype** that can demonstrate the viability of Tajweed mistake detection. The prototype will likely handle a subset of rules robustly and have a basic web interface. This prototype will serve as a proof-of-concept for our QUL grant application – showcasing both the *technical feasibility* and the *educational value* of the idea. In terms of next steps beyond the prototype, we foresee: expanding to more Tajweed rules (like advanced madd types, rules of starting and stopping, etc.), refining the model with more data (perhaps through user feedback loops), and possibly implementing the solution on mobile devices for broader reach.

Ultimately, this project is more than just a technical endeavor; it's about enhancing the way the Quran is taught and practiced in the modern age. By providing instant, personalized feedback, we lower the barrier for students to improve their recitation and memorize with correct Tajweed. We also provide a tool for teachers to supplement their sessions, potentially allowing them to assign practice that the AI can supervise. The positive implications range from individual convenience (a student practicing late at night can get feedback without a teacher present) to global impact (standardizing Tajweed learning for remote or underserved communities).

In conclusion, the Tajweed AI web application aims to embody a successful fusion of **cutting-edge AI** with **classical Islamic education**. The project's comprehensive planning – from data and models to UI and ethics – sets the stage for a compelling prototype. We are confident that with diligence and the timeline outlined, we will deliver a system that impresses both technically and in its service to the Quranic educational mission. With the support of initiatives like the QUL grant and guidance from organizations like Tarteel, this prototype could evolve into a full-fledged feature that benefits Quran learners worldwide, and in doing so, it will demonstrate our team's capability and passion for innovating in the Islamic tech space.

[8] [5] [7] [12]

---

[1] [2] [3] [10] [11] [12] [13] Mispronunciation Detection of Basic Quranic Recitation Rules using Deep Learning | Papers With Code
https://paperswithcode.com/paper/mispronunciation-detection-of-basic-quranic

[4] [8] Tarteel's ML Journey: Part 1 - Intro & Data Collection
https://tarteel.ai/blog/tarteels-ml-journey-part-1-intro-data-collection/

[5] [6] [7] [17] Introducing Mistake Detection
https://tarteel.ai/blog/introducing-mistake-detection/

[9] Tarteel A.I Launches QUL - The Quranic Universal Library
https://tarteel.ai/blog/qul-launch/

[14] arxiv.org
https://arxiv.org/pdf/2405.02675

[15] [16] RIVA Conformer ASR Arabic | NVIDIA NGC
https://catalog.ngc.nvidia.com/orgs/nvidia/teams/tao/models/speechtotext_ar_ar_conformer

18 Tarteel: Your A.I. Companion for Quran Study | Deepgram
https://deepgram.com/ai-apps/tarteel

19 arxiv.org
https://arxiv.org/pdf/2305.06429