1. Idea:
   1. **Initial idea**: Quran web system, it's main idea is helping user memorize the Quran without any machine learning approaches.
   2. **Added features**: add recitation authentication method, using AI and Deep Learning to avoid the need for the human element.
      1. ***Method 1***: use ASR (Automatic Speech Recognition), then compare resulting text to the Quran
         1. ***Pros***: availability; a lot of models and resources are available for such approach whether it be from sites like [HuggingFace](https://huggingface.co/) or Mozilla's open-source Deepspeech engine.
         2. **Cons:** doesn't accurately evaluate the user's pronunciation (التشكيل، التجويد)
      2. ***Method 2***: depend entirely on Audio; we train a model on an Aya and measure the distance between the user's recitation with the Imam's.
         1. ***Pros***: accurately determine the accuracy of the user's recitation by comparing it to a correct reference.
         2. **Cons:** 
            1. Higher risk, asthe method isn't used in any application we know (not guaranteed to work)
            2. With the small dataset we have at the moment, the model might decline correct recitations, which might be frustrating to the user.
2. Data sets:
   1. [**Kaggle's competition Data set:**](https://www.kaggle.com/competitions/quran-asr-challenge/data) we found this data set as a part of a competition to make an ASR model for the Quran.  
      **dataset description:**

The dataset contains the Imams' recitations in the form of mp3 files, also contains a csv file with the mp3 file name and the text corresponding to the Aya

* + 1. Pros:

the dataset is quite large (containing 86055 files) which makes it perfect to use if we choose the ASR model approach.

* + 1. Cons:

1. The Dataset is meant for a competition, which is why 15,000 audio file of the data doesn't have labels (Aya text) in the CSV file, but the main issue with this is **we don't have the complete Quran**, as the missing 15,000 files in the test set is unusable.
2. The Arabic text (with tashkeel) in the csv file is not possible to search using neither google sheets nor Excel, unless you copy the Aya from the CSV file itself, which makes it difficult to navigate and explore the dataset.
   1. [**Tarteel's Users' Dataset and 7 Imams recitation:**](https://archive.org/details/quran-speech-dataset) we found an archive containing some of tarteel's users' dataset (around 12 filtered users) reciting each Aya, with also 7 Imams' recitation of the full Quran

Content:

\* 7 Imam Full Quran Recitation: 7\*6236 wav file

- csv contains the Text info for 11k subset short wav file

\* Tarteel.io user dataset ~25k wav

- csv contains the Text info for 18k subset of the accepted user quality

* 1. Final Data set: we used mixed parts from each Dataset for 2 selected Suras (الفلق) and (الناس) and decided to further proceed with (الناس).

1. Searching how the "Tarteel" App worked:

We read a published [Engineering blog](https://blog.tarteel.ai/tarteels-ml-journey-part-1-intro-data-collection/) from Tarteel about how they did their preprocessing pipeline, using the Django framework to fit the data they collected to their needs. Also found [their Data collection site's Backend repository on github](https://github.com/AliOsamaHassan/tarteel-api) but decided to not further proceed with their methods as we settled on using NodeJS as the Backend framework.

1. Searching about how to deal with Audio and Text to use them in DL.
   1. Audio: we did our research about how to extract features from an Audio file:
      1. Turn the Audio file into a Spectogram
      2. Mel-Spectogram
      3. MFCC
2. Pretrained Model:
   1. [DeepSpeech-Quran:](https://github.com/tarekeldeeb/DeepSpeech-Quran)

A model trained on the Quran as a dataset (by [Eng. Tarek Eldeeb](https://github.com/tarekeldeeb/)), he used a Speech Engine by Mozilla called “DeepSpeech” trained it on the whole Holy Quran creating an ASR model.

The model can simply recognize Quran texts (with tashkeel) and may also output time tags for each word. Several integration examples may be found in the parent

From that model's GitHub repository, we found more data and added them to the dataset we gathered,

* 1. [Mozilla’s DeepSpeech Engine](https://github.com/mozilla/DeepSpeech)

DeepSpeech is an open-source Speech-To-Text (STT) engine, using a model trained by machine learning techniques based on Baidu's Deep Speech research paper, its core architecture is a Bi-RNN (Bi-Directional Recurrent Neural Network).

Project DeepSpeech uses Google's TensorFlow to make the implementation easier.

It can run in real time on devices ranging from a Raspberry Pi 4 to high power GPU servers, also has integrations with multiple frameworks such as NodeJS.

1. Searched about Neural Networks architecture, and which one is best to use for our project, and concluded that:

CNNs work best for spatial data and thus is the most suitable option for image and video processing. RNN, on the other hand, work on sequential data and thus proves to be an appropriate option for text and speech analysis, however we plan on using Spectrograms (images) as features so we might lean towards using the CNN architecture.

1. Coding:
   1. Preprocessing:
      1. We applied what we learned about spectrograms and Mel-spectrograms and MFCCs, extracted each one for a selected Aya and

Tools:

* Front end:
  + Figma for the UI
  + HTML
  + CSS
  + JS
  + Bootstrap
  + ReactJS
* ML:
  + Python
    - Librosa
    - Scikit learn
  + TensorFlow
    - Keras
  + Kaggle
* Back end:
  + NodeJS
  + MongoDB
  + Postman
  + Express

**Roles:**

ML:

1. Abdelrahman 3arfa
2. Habeba Hossam

Front End:

1. Ashraqat Ahmed
2. Mei Habib

Back End:

1. Ahmed Sallam
2. Tamer Medhat