

# Fiabilité/Confiance de transcription" (avec AssemblyAI)

✿ Statut	A faire
📅 Date de livraison	@October 7, 2025
⌚ Priorité	🔥🔥🔥

Cahier des charges — "Fiabilité (V1)"

Objectif

Données côté backend (Nico)

Affichage (Front — vue "Éditeur")

Règles (sans regex, sans risque)

Interaction audio

États & erreurs

KPIs (tracking)

"Et si les timestamps sont imparfaits ?"

Roadmap ultra courte

Mini-pseudo-code front (vue "Éditeur")

Réponses rapides aux doutes de Nico

Où ça colle avec notre doc backend

Untitled

## Cahier des charges — "Fiabilité (V1)"

### Objectif

Permettre à l'utilisateur de **trouver et écouter** les passages possiblement douteux **sans modifier** le texte, que l'on ait des horodatages **mots, phrases** ou seulement **blocs**.

# Données côté backend (Nico)

1. Transcription AssemblyAI avec confidences activées, et récupérer :

- Idéal : mots horodatés [{text,start,end,confidence}]
- Toléré : phrases horodatées [{text,start,end,words:[...]}]
- Dégradé : paragraphe/locuteur [{speaker,start,end,text,avg\_conf}]

2. Exposez dans `receiveText` un champ `confidencePayload` non transformé :

```
{  
  "confidencePayload": {  
    "type": "sentences" | "words" | "paragraphs",  
    "items": [ ... ] // JSON brut renvoyé par AssemblyAI ou agrégé  
  }  
}
```

Information manquante dans README.html : la spec exacte de ce champ n'existe pas encore → à ajouter. L'endpoint `receiveText` existe déjà.

3. Ne rien changer au texte final : le JSON de confiance vit à côté (pas d'injection `[inaudible]` en V1).

4. Endpoints utilisés et déjà documentés :

- `receiveText` (récupération du transcript)
- `getJobsInfo` (listing)
- (option V2) **Word Boost 2** : `setWordBoostList2`, `getStatusWordBoost2`, `getWordBoost2` pour enrichir le vocabulaire métier.

Si aucun timestamp n'est dispo : afficher quand même un score global + une liste de mots faibles (sans bouton "Écouter"). Mode fallback.

## Affichage (Front — vue "Éditeur")

- En-tête (au-dessus du panneau droit) :

"Fiabilité : 82 %" + sélecteur **Niveau** (Précis=0,90 / Standard=0,80 / Large=0,70).

- **Bloc "À vérifier"** (dans la colonne droite) :  
liste paginée d'**alertes**. Chaque alerte affiche :
  - *Timestamp* (si dispo) + *extrait* de la phrase,
  - **mots faibles** (texte + score),
  - boutons ► **Écouter 2 s** (si timestamp), **Marquer OK, + Dictionnaire** (Word Boost v2 – V2).
- **Surlignage inline (option)** : uniquement si on a les mots horodatés → `<mark>` non destructif dans la zone centrale. Sinon, on reste **liste + écoute**.

## Règles (sans regex, sans risque)

- **Score global** : moyenne des confidences (pondérée par durée/nombre de mots).
- **Alerte si** :
  - $\geq 1$  **mot** < seuil (**mode words**),
  - ou **phrase** contenant  $\geq 1$  mot < seuil (**mode sentences**),
  - ou **bloc** avec moyenne < seuil (**mode paragraphs**).
- **Regrouper** les mots faibles contigus en **segment  $\leq 5$  s** pour l'écoute.
- **Jamais** de `replace()` du texte. Le surlignage est un **overlay**, pas une réécriture.

## Interaction audio

- Bouton ► **Écouter** : seek à `max(0, start-1000ms)` puis lecture jusqu'à `end+1000ms`.
- Si timestamp **phrase** uniquement : seek **début de phrase** (OK pour V1).
- Si timestamp **paragraphe** uniquement : seek **début du bloc** (dégradé, mais utile).

## États & erreurs

- `confidencePayload` absent → bannière : “**Fiabilité non disponible pour ce document**”.
- Réseau/serveur → message clair + “Réessayer”.
- A11y : `role="list"` , `aria-label` sur alertes, boutons accessibles clavier.

## KPIs (tracking)

- `avg_conf` , `low_conf_count` , `alerts_count` , `%alerts_with_ts` , temps Écouter → Marquer OK.
- 

## “Et si les timestamps sont imparfaits ?”

- **OK en V1** : on ne touche pas le texte; on se contente de **pointer** une **phrase** ou un **bloc** et d'offrir l'**écoute**.
  - **V1.5 (option)** : heuristique d'alignement pour surbriller approximativement un extrait même sans `words` .
  - **V2 (option)** : marqueurs `[inaudible]` **après validation utilisateur + Word Boost** en un clic.
- 

## Roadmap ultra courte

- **Semaine 1 (V1)** : `confidencePayload` exposé + liste d'alertes avec écoute; seuils 0.90/0.80/0.70; score global.
  - **Semaine 2 (V1.5 option)** : surlignage inline **si** mots horodatés.
  - **V2 (option)** : `[inaudible]` sur validation + **Word Boost 2 UI** et export annexe.
- 

## Mini-pseudo-code front (vue “Éditeur”)

```
function buildAlerts(confidencePayload, threshold) {
  if (!confidencePayload) return [];
  if (confidencePayload.type === 'words') {
    const words = confidencePayload.items; // [{text,start,end,confidence}]
```

```

const alerts = []; let cur=null;
for (const w of words) {
  if (w.confidence < threshold) {
    cur ? (cur.end=w.end, cur.words.push(w)) : (cur={start:w.start,end:w.end,
words:[w]});
  } else if (cur) { alerts.push(cur); cur=null; }
}
if (cur) alerts.push(cur);
return alerts.map(a => ({
  start: a.start, end: a.end,
  text: a.words.map(w=>w.text).join(' '),
  lows: a.words.map(w=>({text:w.text, score:w.confidence}))
}));
}
if (confidencePayload.type === 'sentences') {
  return confidencePayload.items
    .filter(s => s.words.some(w => w.confidence < threshold))
    .map(s => ({
      start: s.start, end: s.end, text: s.text,
      lows: s.words.filter(w => w.confidence < threshold)
        .map(w => ({text:w.text, score:w.confidence}))
    }));
}
if (confidencePayload.type === 'paragraphs') {
  return confidencePayload.items
    .filter(p => p.avg_conf < threshold)
    .map(p => ({ start:p.start, end:p.end, text:p.text, lows:[] }));
}
return [];
}

```

## Réponses rapides aux doutes de Nico

- “Est-ce faisable côté provider ?” → Oui, AssemblyAI fournit **confiance + timestamps** (mots et/ou phrases). On **ne remplace rien** en V1, on **affiche** et on **fait écouter**.
  - “Et si on n'a pas les mots horodatés ?” → On fonctionne en **mode phrases ou paragraphes** : liste + lecture ciblée (pas de surlignage).
  - “Risque de remplacer tous les ‘merci’ ?” → **Aucun** : V1 n'utilise **aucune regex** ni remplacement.
- 

## Où ça colle avec notre doc backend

- `receiveText` = endpoint de récupération du transcript (on y ajoute `confidencePayload` ).
- `getJobsInfo` = listing (pour afficher l'état “Fiabilité dispo / non dispo”).
- **Word Boost 2 (V2)** : `setWordBoostList2` , `getStatusWordBoost2` , `getWordBoost2` déjà décrits.

Information manquante dans README.html à ajouter par Nico :

- le flag d'entrée côté transcription (si besoin) pour demander confidences/phrases,
- la structure exacte de `confidencePayload` renvoyée par `receiveText` .