Intégration API & Dashboard — Plan complet + Suggestions

Voici comment présenter clairement **comment tout ce que vous avez développé sera intégré** dans un dashboard admin.

---

1 Architecture finale côté API (ce que VOUS fournissez)

Votre équipe IA fournit au dashboard :

1. API d'analyse de contenu

Routes principales :

Endpoint	Méthode	Description
/predict/sentiment	POST	Retourne sentiment + score
/predict/toxicity	POST	Retourne toxicité, insultes, haine
/predict/misinformation	POST	Retourne classification fake news / neutre
/predict/politics	POST	Détection de contenu politique
/analysis/full	POST	Lance tous les modèles → analyse complète
/stats/models	GET	Retourne stats globales des modèles (activité, charge, etc.)
/stats/logs	GET	Retourne les logs d'analyses

2. Système de logs

Chaque analyse est sauvegardée avec :

texte soumis

résultats

timestamp

utilisateur (si applicable)

origine (API / dataset / batch)

Ces logs servent **au dashboard** pour afficher des stats.

3. Système de monitoring simple

Modules que vous exposez :

état des microservices

disponibilité (UP/DOWN)

temps moyen d'inférence

volume des requêtes

erreurs

Exposé via :

```
/health
/metrics
```

---

2 Ce que LE BACKEND DU DASHBOARD doit prévoir

A. Appeler vos modèles via API REST

Il doit :

envoyer le texte à analyser

recevoir la réponse JSON

stocker les résultats dans une base interne ou les afficher directement

Exemple d'appel :

```
POST /predict/toxicity
```

```
{
  "text": "Ton travail est nul."
}
```

Réponse :

```
{
  "toxicity": 0.87,
  "hate": false,
  "insult": true}
```

B. Rafraîchissement automatique

Pour suivre les stats en temps réel :

pooling (toutes les 3 secondes)

ou websockets (optionnel)

C. Sécurisation API

Le camarade doit ajouter :

Un token JWT pour sécuriser l'accès

Vérifier limites (rate limiting)

Mettre un rôle admin pour accéder au dashboard

D. Stockage des logs & historiques

Deux options :

Dashboard stocke tout → PostgreSQL / Mongo

Vos services IA stockent, dashboard lit → fichiers JSON / SQLite

1.

Recommandation :

Centraliser dans le backend du dashboard pour simplifier.

---

3 Ce que LE FRONTEND DU DASHBOARD doit préparer

Pages recommandées :

1. Page d'analyse manuelle

Champ texte + bouton → affiche :

sentiment

toxicité

chart radar des scores

tags (haine, toxique, fake news…)

2. Page "Historique des analyses"

Tableau montrant :

texte analysé

résultats

date

bouton "voir détails"

3. Page Monitoring IA

Basée sur vos endpoints `/health` et `/metrics` :

statut des modèles (en ligne / hors ligne)

temps moyen d’inférence

charge du CPU/GPU (optionnel)

courbes de trafic (nombre d’analyses / heure)

4. Page Statistiques

Graphiques :

% de toxicité par jour

Répartition des sentiments

Nombre de fake news détectées

Utilisation des modèles

---

4 Suggestions pour faciliter l’intégration (ce que vous proposez à votre coéquipier)

Voici que vous pouvez écrire au camarade :

Proposition 1 — Standardisation des réponses JSON

Exemple de réponse *globale* :

```
{
  "sentiment": {
    "label": "positive",
    "score": 0.92
  },
  "toxicity": {
    "score": 0.14,
    "hate": false,
    "insult": false
```

```
  },
  "misinformation": {
    "label": "neutral"
  },
  "politics": {
    "is_political": true,
    "category": "campagne"
  },
  "timestamp": "2025-11-15T18:39:00Z"}
```


Ainsi le dashboard peut afficher les données facilement.

Proposition 2 — Fournir un fichier OpenAPI/Swagger

Votre équipe peut fournir :

`openapi.yaml`

Pour que le camarade génère automatiquement :

- routes
- clients API
- modèles

Proposition 3 — Ajouter un mode “simulation rapide”

Pour tester l’UI sans attendre les modèles IA :

`/predict/mock`

Retourne une réponse fixe → utile pour mettre au point le frontend avant votre code IA final.

Proposition 4 — Ajouter un endpoint batch

Permet au dashboard d'envoyer plusieurs textes d'un coup :

```
POST /analysis/batch
[
  {"text": "..."},
  {"text": "..."}
]
```

---

5 Résumé final (facile à mettre dans votre rapport)

Voici un résumé très clair :

Objectif du module "Intégration Dashboard"

Même si le dashboard n'est pas développé par notre équipe, nous définissons comment nos micro-services IA s'intègrent à l'interface admin.

Nous fournissons :

- APIs REST (sentiment, toxicité, fake news, politique)
- Endpoint d'analyse globale
- Système de logs
- Endpoints monitoring (health + metrics)

Le backend dashboard doit :

- appeler nos APIs
- stocker ou afficher les résultats
- gérer accès sécurisé (JWT + rôles)

compiler des statistiques à partir des logs

`Le frontend dashboard doit :`

proposer une interface d'analyse de texte

afficher résultats sous forme de graphes et tags

montrer l'historique

afficher état des modèles (UP/DOWN)

suivre l'activité des microservices

`Nous proposons aussi :`

schéma JSON standardisé

fichier OpenAPI

endpoint mock pour tests

possibilité d'analyse batch

EN RESUME:
Salut, pour l'intégration de notre partie IA au dashboard, voici comment on propose de procéder. Même si tu es en charge du frontend et backend du dashboard, il est important de prévoir comment nos micro-services IA communiqueront avec ton interface.

Nous fournirons des **API REST** qui permettent au dashboard d'envoyer du texte à analyser et de recevoir les résultats sous forme de JSON standardisé. Les routes principales incluent l'analyse de la **toxicité, haine, fake news, contenu adulte**, ainsi qu'un endpoint global qui lance tous les modèles en une seule requête. Chaque analyse sera enregistrée avec le texte soumis, les résultats, le timestamp et l'origine (batch, API, dataset), ce qui permettra au dashboard de générer des statistiques ou de l'historique. Nous prévoyons également un endpoint de monitoring (`/health` et `/metrics`) pour indiquer l'état des modèles, leur disponibilité, le temps moyen d'inférence et le volume des requêtes.

Du côté **backend du dashboard**, il faudra que l'application puisse appeler nos APIs, récupérer les réponses JSON et les stocker dans une base (PostgreSQL ou MongoDB)

ou les afficher directement. Il sera utile d'avoir un rafraîchissement automatique (via pooling ou websockets) pour suivre les analyses en temps réel. L'accès aux endpoints doit être sécurisé via un token JWT et éventuellement des rôles pour les utilisateurs.

Pour le **frontend**, nous recommandons plusieurs pages : une page d'analyse manuelle où un utilisateur peut saisir un texte et voir les résultats avec des tags et graphiques, une page historique affichant toutes les analyses effectuées, une page statistiques montrant la répartition des toxicités, fake news et sentiments, et enfin une page monitoring affichant l'état des modèles et les performances des micro-services.

Pour faciliter ton travail, nous proposons d'utiliser un **format JSON standardisé** pour toutes les réponses, un fichier OpenAPI/Swagger pour générer automatiquement les routes et modèles, un endpoint `mock` pour simuler des réponses lors du développement de l'UI, et un endpoint batch pour analyser plusieurs textes simultanément.

En résumé, notre objectif est que tu puisses intégrer facilement notre partie IA dans le dashboard, afficher les résultats en temps réel, gérer l'historique et le suivi des modèles, et préparer des visualisations pertinentes pour le CSC. Tout est conçu pour que nos services restent indépendants et modulables, et que le dashboard puisse évoluer sans modifier nos modèles IA.