Cahier des Charges : ScribeCoach

1. Introduction et Objectifs du Projet

- **Nom de l'application :** ScribeCoach
- **Version :** 2.0 (refonte)
- **Objectif principal :** Fournir une plateforme web interactive et pédagogique permettant aux collégiens et lycéens d'améliorer leurs compétences en expression écrite grâce à un feedback personnalisé généré par une IA (Google Gemini). L'application doit également offrir aux enseignants des outils pour créer des sujets et suivre les progrès des élèves.
- **Principes directeurs :** Autonomie de l'élève, feedback constructif, gamification engageante, facilité d'utilisation pour élèves et enseignants, design responsive et accessible.

2. Public Cible

- **Utilisateurs principaux :**
    - **Élèves :** Collégiens (6ème à 3ème) et Lycéens (Seconde à Terminale).
    - **Enseignants :** Professeurs de Français ou de matières nécessitant une forte compétence rédactionnelle.
- **Contexte d'utilisation :** En classe (avec supervision) ou à la maison (en autonomie).

3. Rôles Utilisateurs et Permissions

- **3.1. Rôle "Élève"**
    - S'identifier (pour cette version, profil unique pré-chargé).
    - Accéder à un tableau de bord personnalisé (progression, points, niveaux, badges, historique récent).
    - Consulter et sélectionner des sujets d'écriture proposés par les enseignants.
    - Rédiger et soumettre des textes en réponse aux sujets.
    - Recevoir une analyse détaillée de son texte par l'IA (forme, fond, structure, style, points positifs, pistes d'amélioration, score conceptuel).
    - Consulter l'historique de ses soumissions et des feedbacks associés.
    - Améliorer un texte déjà soumis en se basant sur le feedback.
    - Changer de rôle (pour passer en mode Enseignant si besoin, via rechargement de l'app).
- **3.2. Rôle "Enseignant"**
    - S'identifier (pour cette version, accès direct au portail enseignant).
    - Accéder à un tableau de bord agrégeant les données de soumissions des élèves (statistiques, tendances).
    - Gérer les sujets d'écriture :
        - Créer de nouveaux sujets (titre, description/consigne, niveau scolaire, mots-clés spécifiques pour orienter l'IA).

        - Modifier des sujets existants.
        - Supprimer des sujets (avec avertissement sur l'impact potentiel).
        - Visualiser la liste des sujets.
    - Consulter les textes soumis par les élèves et le feedback IA correspondant.
    - Changer de rôle (pour passer en mode Élève si besoin, via rechargement de l'app).

4. Fonctionnalités Détaillées

- **4.1. Module Commun / Global**
    - **Page d'Accueil / Sélection de Rôle :**
        - Présentation succincte de ScribeCoach.
        - Boutons clairs pour sélectionner le rôle "Élève" ou "Enseignant".
    - **Structure de Page (Layout) :**
        - **En-tête (Header) :** Logo et nom "ScribeCoach", nom du mode actuel (Élève/Enseignant), bouton pour changer de rôle (provoquant un rechargement de l'application vers la page de sélection).
        - **Pied de page (Footer) :** Copyright, informations de développement (ex: "Développé par FTLV-ST 2025"), version.
        - **Zone de contenu principale :** Dynamique en fonction du rôle et de la vue.
    - **Notifications / Alertes :** Système d'alertes (succès, erreur, avertissement, information) non-intrusif et accessible.
- **4.2. Espace Élève**
    - **Tableau de Bord Élève :**
        - **Accueil personnalisé :** "Bonjour, [Nom de l'élève]".
        - **Progression :** Niveau actuel, Points XP, barre de progression vers le prochain niveau.
        - **Badges :** Section visuelle affichant les badges obtenus.
        - **Dernières activités :** Résumé des 3 dernières soumissions (sujet, date, score/statut).
        - **Accès rapide :** Bouton pour "Choisir un nouveau sujet".
    - **Sélection de Sujets :**
        - Affichage des sujets sous forme de cartes responsives.
        - Chaque carte doit indiquer : Titre du sujet, Niveau, Description courte.
        - Filtre par niveau (optionnel, pour une V2).
        - Bouton "Commencer à écrire" sur chaque carte.
    - **Espace de Rédaction :**
        - Rappel du sujet choisi (Titre, Description complète, Mots-clés pour l'IA).
        - Zone de texte (textarea) confortable, avec hauteur ajustable ou suffisante.
        - Compteur de mots en temps réel.
        - Boutons d'action : "Soumettre pour analyse", "Annuler". Le bouton "Soumettre" est désactivé si le texte est trop court ou pendant une analyse.
        - En cas de nouvelle tentative sur un texte : pré-remplissage du texte.

        - En cas d'erreur d'analyse précédente : bouton "Réessayer l'analyse".
    - **Affichage du Feedback IA :**
        - Présentation claire et structurée du feedback.
        - **Score Conceptuel :** Mis en évidence (ex: "Maîtrisé", "En Progrès") avec code couleur.
        - **Sections distinctes (potentiellement dans des accordéons/onglets pour lisibilité) :**
            - Résumé Général de l'analyse.
            - Points Forts (liste à puces).
            - Pistes d'Amélioration (chaque piste avec : domaine, commentaire, suggestion concrète).
            - Conseil Personnalisé.
            - Rappel du Texte Soumis par l'élève.
        - **Actions Post-Feedback :**
            - "Améliorer ce texte" (retourne à l'espace de rédaction avec le sujet et le texte).
            - "Écrire un nouveau texte" (retourne à la sélection de sujets).
            - "Retour au tableau de bord".
    - **Gamification :**
        - **Points (XP) :** Attribués pour chaque soumission analysée avec succès. Le nombre de points peut varier (ex: +10 points de base, + points bonus selon le score conceptuel).
        - **Niveaux :** Seuils de points définis pour atteindre de nouveaux niveaux.
        - **Badges :** Récompenses visuelles pour des étapes clés (ex: "Premier texte soumis", "5 textes analysés", "Série de 3 'Maîtrisé'").
- **4.3. Espace Enseignant**
    - **Navigation Enseignant :** Barre de navigation ou onglets clairs pour basculer entre :
        - "Tableau de Bord des Progrès"
        - "Gestion des Sujets"
    - **Tableau de Bord des Progrès :**
        - **Indicateurs Clés :** Nombre total de soumissions, nombre de soumissions analysées, score moyen (global et/ou par niveau), nombre d'élèves actifs (simplifié à "1" pour la version mono-profil élève).
        - **Graphiques :**
            - Évolution du nombre de soumissions par sujet.
            - Distribution des scores conceptuels par sujet.
        - **Liste des Soumissions :**
            - Tableau responsif avec colonnes triables : Date, Élève (ID), Sujet, Statut (Analysé, En attente, Erreur), Score Conceptuel, Score Numérique.
            - Action "Voir Feedback" pour chaque soumission analysée, ouvrant une modale.

- **Gestion des Sujets :**
    - Affichage de la liste des sujets existants sous forme de tableau ou de cartes.
    - Chaque sujet affiche : Titre, Niveau, Description courte, Mots-clés IA.
    - Bouton "Créer un nouveau sujet".
    - Actions par sujet : "Modifier", "Supprimer" (avec modale de confirmation).
- **Formulaire de Création/Édition de Sujet (dans une Modale) :**
    - Champs :
        - Titre (texte, obligatoire).
        - Description / Consigne (textarea, obligatoire).
        - Niveau (texte, ex: "Collège 6ème", "Lycée Première", obligatoire).
        - Mots-clés pour l'IA (texte, optionnel, avec aide sur le format/usage).
    - Boutons : "Enregistrer", "Annuler".
- **Modale d'Aperçu du Feedback (vue enseignant) :**
    - Affiche le feedback IA d'une soumission spécifique, de manière similaire à la vue élève, mais en lecture seule.

5. Intégration de l'IA (Google Gemini API)

- **Modèle d'IA :** `gemini-2.5-flash-preview-04-17` (ou le dernier modèle flash recommandé pour le texte).
- **Clé API :** Fournie **exclusivement** via la variable d'environnement `process.env.API_KEY`. L'application ne doit JAMAIS permettre la saisie de la clé API.
- **Interaction avec l'API :**
    - Utilisation du SDK `@google/genai`.
    - **Prompt envoyé à l'IA :**
        - **Instruction Système :** Définit le rôle de ScribeCoach, l'audience, les axes d'analyse (forme, fond, structure, style), l'importance d'un ton encourageant et pédagogique, et EXIGE une réponse au format JSON strict.
        - **Contenu Utilisateur :** Texte de l'élève, accompagné du contexte du sujet (titre, description, niveau, et les `keywordsForAI` spécifiés par l'enseignant).
    - **Configuration de la requête :** `responseMimeType: "application/json"`, `temperature` (ex: 0.7 pour un équilibre créativité/précision).
    - **Format de Réponse JSON attendu de l'IA (exemple) :**

Generated json

```
{
  "analyseForme": "Analyse détaillée de l'orthographe, grammaire, syntaxe...",
  "analyseFond": "Analyse de la pertinence, argumentation, richesse des idées...",
  "analyseStructure": "Analyse de l'organisation, paragraphes, connecteurs...",
  "analyseStyle": "Analyse du vocabulaire, ton, figures de style...",
  "pointsPositifs": ["Clarté de l'introduction.", "Bon usage des exemples."],
  "pointsAmelioration": [
    { "domaine": "Conjugaison", "commentaire": "Quelques erreurs sur les temps.",
```

```
"suggestion": "Relire la règle sur l'accord du participe passé." }
  ],
  "conseilsPersonnalises": "Pour progresser, concentrez-vous sur la variété des
connecteurs logiques.",
  "scoreConceptuel": "En Progrès" // (Débutant, En Progrès, Maîtrisé, Expert)
}
```

content_copydownload

Use code with caution.Json

- **Traitement de la Réponse :**
  - Nettoyage de la chaîne JSON (suppression des `json ...` si présents).
  - Parsing du JSON.
  - Mapping des champs de la réponse IA vers la structure `AIFeedback` utilisée par l'UI.
  - Gestion robuste des erreurs API et de parsing.

6. Gestion des Données

- **Support de stockage :** `localStorage` du navigateur.
- **Structure des données (Types TypeScript) :**
  - `UserRole`: `student`, `teacher`.
  - `Subject`: `id`, `title`, `description`, `level`, `keywordsForAI?`.
  - `AIFeedback`: `overallSummary`, `strengths[]`, `areasForImprovement[]`, `personalizedTip`, `conceptualScore`.
  - `StudentSubmission`: `id`, `studentId`, `subjectId`, `subjectTitle`, `text`, `feedback?`, `score?` (numérique, ex: 0-100), `timestamp`, `status` (`pending`, `analyzed`, `error`).
  - `StudentProfile`: `id` (fixe pour cette version), `name`, `level`, `points`, `badges[]`, `submissions[]` (liste des IDs de soumission ou objets complets synchronisés).
- **Clés `localStorage` :** Définies dans `constants.ts` (ex: `SCRIBECOACH_SUBJECTS`, `SCRIBECOACH_SUBMISSIONS`, `SCRIBECOACH_PROFILE`).
- **Initialisation des données :** Fournir un jeu de sujets initiaux si `localStorage` est vide pour les sujets enseignants. Profil élève initial créé si inexistant.
- **Service de données (`dataService.ts`) :** Abstraire les opérations CRUD sur `localStorage`.

7. Conception UI/UX et Exigences Techniques

- **7.1. Interface Utilisateur (UI)**
  - **Design :** Moderne, épuré, intuitif et engageant. Esthétique professionnelle adaptée à un contexte éducatif.
  - **Palette de couleurs :** Harmonieuse (dominante bleue/ciel suggérée, avec des couleurs d'accentuation pour les actions, alertes, et éléments de gamification).

    - **Typographie :** Polices sans-serif lisibles (ex: Inter, Open Sans) avec une hiérarchie claire.
    - **Icônes :** Utilisation d'icônes SVG (ex: Heroicons) pour améliorer la compréhension et l'attrait visuel.
    - **Cohérence :** Maintenir une charte graphique et des patterns d'interaction cohérents à travers toute l'application.
- **7.2. Expérience Utilisateur (UX)**
    - **Navigation :** Claire, simple et prévisible.
    - **Feedback Utilisateur :** Retours visuels immédiats pour les actions (états de chargement, confirmations, erreurs).
    - **Performance perçue :** Optimiser les chargements, utiliser des indicateurs de chargement pour les opérations asynchrones.
    - **Réduction de la charge cognitive :** Présenter l'information de manière concise et progressive. Utiliser des éléments comme les accordéons pour gérer la densité d'information (ex: feedback détaillé).
- **7.3. Responsivité**
    - L'application **doit** être entièrement responsive et s'adapter fluidement à différentes tailles d'écran :
        - Ordinateurs de bureau (large, moyen)
        - Tablettes (portrait, paysage)
        - Smartphones (portrait, paysage)
    - Utilisation de Tailwind CSS pour une approche "mobile-first" ou "desktop-first" cohérente.
    - Test sur des émulateurs et appareils réels.
- **7.4. Accessibilité (A11y)**
    - Objectif : Conformité WCAG 2.1 Niveau AA.
    - HTML sémantique.
    - Navigation clavier complète et logique.
    - Attributs ARIA appropriés pour les composants dynamiques (modales, boutons avec icônes seules, etc.).
    - Contrastes de couleurs suffisants pour le texte et les éléments d'interface.
    - Alternatives textuelles pour les images et icônes porteuses d'information.
    - Labels explicites pour les champs de formulaire.
- **7.5. Performance**
    - Optimisation du code (React, JavaScript).
    - Chargement asynchrone des composants si nécessaire (pour les grosses sections).
    - Minimisation des requêtes API.
    - Utilisation efficiente de `localStorage`.
- **7.6. Compatibilité Navigateurs**
    - Support des deux dernières versions majeures de Chrome, Firefox, Safari, Edge.
- **7.7. Sécurité**

- **Clé API Gemini :** Traitée comme indiqué (variable d'environnement, jamais côté client).
- Pas de XSS : S'assurer que les données utilisateur (textes soumis) sont correctement échappées si affichées directement (React le fait par défaut pour le contenu JSX).

8. Stack Technologique (Recommandée)

- **Langage :** TypeScript
- **Framework/Bibliothèque Frontend :** React (dernière version stable, avec Hooks)
- **Styling :** Tailwind CSS (dernière version stable)
- **Client API Gemini :** `@google/genai` (dernière version stable)
- **Gestion d'état (si besoin au-delà des hooks React) :** Zustand ou React Context (pour une complexité modérée).
- **Graphiques (pour dashboard enseignant) :** Recharts
- **Bibliothèque d'icônes :** Heroicons (ou similaire, SVG de préférence)
- **Outils de développement :** Vite (pour le build et le serveur de dev), ESLint, Prettier.

9. Structure du Projet (Suggestion)

Generated code

```
/src
  /components
    /common         # Boutons, Modales, Layout, Header, Footer, Icônes, Alertes...
    /student        # Composants spécifiques à l'espace élève
    /teacher        # Composants spécifiques à l'espace enseignant
  /constants        # Constantes de l'application (clés localStorage, seuils, etc.)
  /hooks            # Hooks personnalisés (ex: useLocalStorage)
  /services
    dataService.ts  # Logique pour localStorage
    geminiService.ts# Logique pour l'API Gemini
  /styles           # Styles globaux (si Tailwind ne suffit pas, ex: base styles)
  /types            # Définitions des types et interfaces
  /utils            # Fonctions utilitaires
  App.tsx           # Composant racine, gestion des rôles principaux
  main.tsx          # (ou index.tsx) Point d'entrée React
/public
  index.html
  # ... autres assets statiques
metadata.json       # (si requis par la plateforme de déploiement)
```

content_copydownload

Use code with caution.

10. Livrables Attendus (pour cette refonte)

- Code source complet de l'application ScribeCoach.
- Application fonctionnelle respectant toutes les spécifications ci-dessus.
- Documentation minimale du code (commentaires JSDoc pour les fonctions et composants complexes).

11. Évolutions Futures (Hors périmètre de cette version)

- Authentification multi-utilisateurs (élèves et enseignants).
- Base de données backend pour la persistance des données.
- Fonctionnalités de classe (regroupement d'élèves).
- Assignation de sujets spécifiques à des groupes d'élèves.
- Export des données pour les enseignants.
- Tableau de bord plus détaillé pour le suivi individuel des compétences par élève.