Voici un \*\*cahier des charges\*\* détaillé pour le projet \*\*"EduRédact"\*\*, destiné aux développeurs. Ce document décrit les fonctionnalités, les technologies à utiliser, les exigences techniques et les livrables attendus.

---

### \*\*Cahier des Charges : Plateforme EduRédact\*\*

---

### \*\*1. Contexte et Objectifs\*\*

\*\*EduRédact\*\* est une plateforme en ligne qui propose des services de \*\*traduction\*\*, \*\*rédaction\*\*, \*\*correction\*\* et \*\*relecture\*\* de documents académiques. L'objectif est de fournir aux étudiants et jeunes professionnels un outil simple et efficace pour améliorer la qualité de leurs travaux écrits.

---

### \*\*2. Public Cible\*\*

- \*\*Étudiants\*\* : Universitaires, élèves du secondaire, chercheurs.

- \*\*Professionnels\*\* : Jeunes diplômés rédigeant des CV, lettres de motivation, ou articles.

- \*\*Écoles et Universités\*\* : Partenariats pour offrir des services groupés.

---

### \*\*3. Fonctionnalités Requises\*\*

#### \*\*3.1. Interface Utilisateur (Frontend)\*\*

- \*\*Page d'Accueil :\*\*

- Présentation des services (traduction, rédaction, correction, relecture).

- Formulaire de soumission de documents (type de service, délai, langue, etc.).

- Téléchargement du document à traiter.

- Estimation automatique du prix et du délai.

- \*\*Espace Client :\*\*

- Inscription/Connexion (email, numéro de téléphone, réseaux sociaux).

- Tableau de bord pour suivre les commandes en cours.

- Messagerie pour communiquer avec le rédacteur/traducteur.

- Téléchargement du document final.

- Historique des commandes.

- \*\*Espace Rédacteurs/Traducteurs :\*\*

- Inscription/Connexion (avec vérification des compétences).

- Tableau de bord pour gérer les commandes.

- Système de notation et de feedback pour garantir la qualité.

- \*\*Page de Paiement :\*\*

- Intégration de Mobile Money (Orange Money, MTN Mobile Money).

- Option pour les cartes bancaires (si possible).

- Suivi des paiements et facturation.

#### \*\*3.2. Backend\*\*

- \*\*Gestion des Utilisateurs :\*\*

- Authentification (email, téléphone, réseaux sociaux).

- Rôles (client, rédacteur/traducteur, administrateur).

- \*\*Gestion des Commandes :\*\*

- Soumission, suivi et traitement des commandes.

- Attribution automatique ou manuelle des commandes aux rédacteurs/traducteurs.

- Notifications par email ou SMS (nouvelle commande, avancement, etc.).

- \*\*Gestion des Documents :\*\*

- Stockage sécurisé des documents (cloud ou serveur local).

- Historique des documents traités.

- \*\*Gestion des Paiements :\*\*

- Intégration des APIs de paiement (Mobile Money, cartes bancaires).

- Suivi des transactions et génération de factures.

- \*\*Administration :\*\*

- Tableau de bord administrateur pour gérer les utilisateurs, les commandes et les paiements.

- Rapports et statistiques (nombre de commandes, revenus, etc.).

---

### \*\*4. Technologies Recommandées\*\*

- \*\*Frontend :\*\*

- Framework : \*\*React.js\*\* ou \*\*Vue.js\*\* (pour une interface dynamique et réactive).

- Design : \*\*Bootstrap\*\* ou \*\*Tailwind CSS\*\* (pour un design responsive et moderne).

- \*\*Backend :\*\*

- Langage : \*\*Node.js\*\* (avec Express.js) ou \*\*Python\*\* (avec Django/Flask).

- Base de données : \*\*PostgreSQL\*\* ou \*\*MongoDB\*\* (pour la gestion des utilisateurs et des commandes).

- Stockage des fichiers : \*\*AWS S3\*\* ou \*\*Firebase Storage\*\* (pour le stockage sécurisé des documents).

- \*\*Authentification :\*\*

- \*\*Firebase Authentication\*\* ou \*\*Auth0\*\* (pour une gestion simple et sécurisée des utilisateurs).

- \*\*Paiements :\*\*

- Intégration des APIs de Mobile Money (Orange Money, MTN Mobile Money).

- Option pour Stripe ou PayPal (si les cartes bancaires sont supportées).

- \*\*Notifications :\*\*

- \*\*Twilio\*\* (pour les SMS) et \*\*SendGrid\*\* (pour les emails).

---

### \*\*5. Exigences Techniques\*\*

- \*\*Responsive Design\*\* : La plateforme doit être accessible et fonctionnelle sur tous les appareils (desktop, tablette, mobile).

- \*\*Sécurité\*\* :

- Chiffrement des données sensibles (mots de passe, documents).

- Protection contre les attaques (XSS, SQL injection, etc.).

- \*\*Performance\*\* : Temps de chargement rapide, même avec des documents volumineux.

- \*\*Scalabilité\*\* : La plateforme doit pouvoir gérer un nombre croissant d'utilisateurs et de commandes.

---

### \*\*6. Livrables Attendus\*\*

1. \*\*Site Web Fonctionnel :\*\*

- Page d'accueil avec formulaire de soumission.

- Espace client et rédacteur/traducteur.

- Système de paiement intégré.

2. \*\*Application Mobile (Optionnel) :\*\*

- Développée avec \*\*Flutter\*\* ou \*\*React Native\*\*.

- Fonctionnalités similaires au site web.

3. \*\*Base de Données :\*\*

- Structure bien organisée pour gérer les utilisateurs, les commandes et les documents.

4. \*\*Documentation Technique :\*\*

- Guide d'installation et de configuration.

- Documentation des APIs (si nécessaire).

5. \*\*Tests et Déploiement :\*\*

- Tests fonctionnels et de sécurité.

- Déploiement sur un serveur cloud (AWS, Google Cloud, etc.).

---

### \*\*7. Planning Prévisionnel\*\*

1. \*\*Phase 1 : Conception et Prototypage (2-3 semaines)\*\*

- Maquettes de l'interface utilisateur.

- Structure de la base de données.

2. \*\*Phase 2 : Développement Frontend et Backend (4-6 semaines)\*\*

- Développement des fonctionnalités principales.

- Intégration des APIs de paiement et de notification.

3. \*\*Phase 3 : Tests et Déploiement (2-3 semaines)\*\*

- Tests fonctionnels et corrections.

- Déploiement sur un serveur cloud.

---

### \*\*8. Budget Estimé\*\*

- \*\*Développement :\*\* 1 000 000 à 2 000 000 FCFA (selon l'expérience du développeur).

- \*\*Hébergement et Maintenance :\*\* 100 000 à 200 000 FCFA par an.

- \*\*Marketing :\*\* 200 000 à 500 000 FCFA (campagnes initiales).

---

### \*\*9. Contact\*\*

- \*\*Porteur du Projet :\*\* [Votre Nom]

- \*\*Email :\*\* [Votre Email]

- \*\*Téléphone :\*\* [Votre Numéro]

---

Ce cahier des charges est conçu pour guider les développeurs dans la réalisation de la plateforme \*\*EduRédact\*\*. Il peut être ajusté en fonction des retours et des besoins spécifiques.

Pour bien organiser le projet \*\*EduRédact\*\*, voici une \*\*structure de projet\*\* détaillée. Cette structure permettra de garder une organisation claire et facilitera le développement, la maintenance et l'évolutivité de la plateforme.

---

### \*\*Structure du Projet EduRédact\*\*

---

### \*\*1. Architecture Globale\*\*

Le projet sera divisé en \*\*3 parties principales\*\* :

1. \*\*Frontend\*\* : L'interface utilisateur (site web et application mobile).

2. \*\*Backend\*\* : Le serveur et la logique métier.

3. \*\*Base de Données\*\* : Le stockage des données.

---

### \*\*2. Structure des Répertoires\*\*

#### \*\*2.1. Frontend\*\*

Le frontend sera divisé en deux parties : \*\*site web\*\* et \*\*application mobile\*\*.

- \*\*Site Web :\*\*

```

frontend-web/

├── public/ # Fichiers statiques (images, favicon, etc.)

├── src/

│ ├── components/ # Composants réutilisables (navbar, footer, etc.)

│ ├── pages/ # Pages de l'application (accueil, espace client, etc.)

│ ├── services/ # Services pour les appels API (connexion, commandes, etc.)

│ ├── styles/ # Fichiers CSS ou SCSS

│ ├── App.js # Point d'entrée de l'application

│ └── index.js # Rendu de l'application

├── package.json # Dépendances du projet

└── README.md # Documentation du projet

```

- \*\*Application Mobile :\*\*

```

frontend-mobile/

├── assets/ # Images, polices, etc.

├── lib/

│ ├── components/ # Composants réutilisables

│ ├── screens/ # Écrans de l'application

│ ├── services/ # Services pour les appels API

│ └── main.dart # Point d'entrée de l'application

├── pubspec.yaml # Dépendances du projet

└── README.md # Documentation du projet

```

---

#### \*\*2.2. Backend\*\*

Le backend sera structuré en suivant une architecture modulaire pour une meilleure maintenabilité.

```

backend/

├── config/ # Fichiers de configuration (base de données, clés API, etc.)

├── controllers/ # Contrôleurs pour gérer les requêtes (utilisateurs, commandes, etc.)

├── models/ # Modèles de données (utilisateurs, commandes, documents, etc.)

├── routes/ # Routes de l'API

├── services/ # Logique métier (gestion des paiements, notifications, etc.)

├── middleware/ # Middleware (authentification, validation, etc.)

├── utils/ # Utilitaires (gestion des erreurs, fonctions communes, etc.)

├── app.js # Point d'entrée de l'application

├── server.js # Configuration du serveur

├── package.json # Dépendances du projet

└── README.md # Documentation du projet

```

---

#### \*\*2.3. Base de Données\*\*

La base de données sera structurée en plusieurs collections/tables pour gérer les utilisateurs, les commandes, les documents, etc.

- \*\*Structure de la Base de Données :\*\*

```

- Utilisateurs (Users)

- id

- nom

- email

- téléphone

- mot de passe (hashé)

- rôle (client, rédacteur, admin)

- date d'inscription

- Commandes (Orders)

- id

- client\_id (référence à l'utilisateur)

- rédacteur\_id (référence à l'utilisateur)

- type\_de\_service (traduction, rédaction, correction, etc.)

- document (lien vers le fichier)

- statut (en attente, en cours, terminé)

- prix

- date de création

- date de livraison

- Documents (Documents)

- id

- commande\_id (référence à la commande)

- fichier\_original (lien vers le fichier)

- fichier\_traité (lien vers le fichier)

- date de téléchargement

- Paiements (Payments)

- id

- commande\_id (référence à la commande)

- montant

- méthode (Mobile Money, carte bancaire)

- statut (réussi, échoué, en attente)

- date de paiement

```

---

### \*\*3. Workflow de Développement\*\*

#### \*\*3.1. Outils et Technologies\*\*

- \*\*Versioning :\*\* Git (avec GitHub, GitLab ou Bitbucket).

- \*\*Gestion des Tâches :\*\* Trello, Jira ou Notion.

- \*\*Communication :\*\* Slack ou Discord pour l'équipe.

#### \*\*3.2. Étapes de Développement\*\*

1. \*\*Initialisation du Projet :\*\*

- Créer les dépôts Git (frontend, backend, mobile).

- Configurer les outils de gestion de tâches.

2. \*\*Développement du Backend :\*\*

- Configurer le serveur et la base de données.

- Implémenter les routes et les contrôleurs.

- Tester les APIs avec Postman ou Insomnia.

3. \*\*Développement du Frontend :\*\*

- Créer les pages et les composants.

- Intégrer les appels API.

- Tester l'interface utilisateur.

4. \*\*Intégration des Paiements et Notifications :\*\*

- Intégrer les APIs de Mobile Money.

- Configurer les notifications par email et SMS.

5. \*\*Tests et Déploiement :\*\*

- Tester l'ensemble de la plateforme (fonctionnel, sécurité, performance).

- Déployer sur un serveur cloud (AWS, Google Cloud, etc.).

---

### \*\*4. Documentation\*\*

- \*\*Documentation Technique :\*\*

- Guide d'installation et de configuration.

- Documentation des APIs (avec Swagger ou Postman).

- \*\*Documentation Utilisateur :\*\*

- Guide d'utilisation pour les clients et les rédacteurs/traducteurs.

---

### \*\*5. Maintenance et Évolutivité\*\*

- \*\*Maintenance :\*\*

- Mettre en place un système de monitoring (erreurs, performance).

- Corriger les bugs et améliorer les fonctionnalités existantes.

- \*\*Évolutivité :\*\*

- Prévoir une architecture scalable pour gérer une croissance du nombre d'utilisateurs.

- Ajouter de nouvelles fonctionnalités (chat en direct, suivi en temps réel, etc.).

---

Cette structure de projet est conçue pour être \*\*modulaire\*\*, \*\*scalable\*\* et \*\*facile à maintenir\*\*. Elle permet de séparer clairement les responsabilités et de faciliter la collaboration entre les développeurs.