Nom du Projet : **"Plateforme pour dynamiser le commerce local"**

Contexte

Dans le cadre de la relance des commerces de proximité, une commune souhaite offrir une vitrine digitale pour ses artisans et commerçants. Cette plateforme permettra aux habitants de découvrir les commerces, leurs produits, services, et événements spéciaux. Le projet met en avant les compétences de développeur web tout en ayant une portée sociale et économique.

Objectifs pédagogiques personnalisés

- Découvrir les spécificités des plateformes vitrines (UX/UI centrée sur la simplicité).
- Intégrer des cartes interactives (API Google Maps ou Leaflet).
- Développer des fonctionnalités e-commerce basiques (par exemple, un formulaire de réservation ou de commande).
- Se familiariser avec l'accessibilité numérique (normes WCAG).
- Travailler en équipe avec une approche agile (Scrum ou Kanban).

Fonctionnalités principales adaptées

1. **Page d'accueil :**
    - Slider pour les promotions du moment.
    - Liste des catégories de commerces (artisanat, alimentation, services, etc.).
    - Une carte interactive affichant les commerces participants.
2. **Pour les visiteurs :**
    - Rechercher un commerce par mot-clé ou catégorie.
    - Voir les fiches des commerces (description, horaires, localisation, photos, contact).
    - Découvrir les événements (marchés, promotions, ateliers, etc.).
3. **Pour les commerçants inscrits :**
    - Création et mise à jour de leur fiche commerce (description, galerie d'images, horaires).
    - Ajout d'événements ou de promotions spécifiques.
4. **Administration :**
    - Validation des inscriptions des commerçants.
    - Modération des contenus publiés (photos, textes).
    - Gestion des statistiques (nombre de visites par commerce).

Technologies suggérées pour un projet réaliste

- **Front-end :**
  - HTML5, CSS3 (avec Tailwind ou Bootstrap pour rapidité).
  - JavaScript avec Vue.js, React ou Angular.
  - Intégration d'API Google Maps ou OpenStreetMap.
- **Back-end :**
  - Node.js (Express) ou PHP (Symfony), selon les préférences des apprenants.
  - Authentification JWT ou session.
- **Base de données :**
  - MySQL pour les données relationnelles.
- **Outils collaboratifs :**
  - GitLab pour le suivi de version.
  - Trello/Notion pour organiser les tâches en équipe.

Livrables attendus

1. Une application web responsive avec une version déployée en ligne.
2. Une documentation utilisateur pour les commerçants (comment créer et gérer leur fiche).
3. Un rapport technique incluant les choix technologiques, les maquettes, et le schéma de la base de données.

Pour enrichir le projet

- **Gamification :** Récompenses virtuelles pour les habitants qui visitent plusieurs commerces.
- **Notifications :** Alertes e-mail ou SMS pour les promotions des commerces favoris.
- **Progressive Web App (PWA) :** Transformer le site en une application mobile installable.

Adaptations :

1. Focus sur l'expérience utilisateur (UX/UI)

- Ajouter un mode sombre pour l'interface.
- Proposer un parcours utilisateur optimisé, avec un moteur de recherche intelligent (suggestions auto, tags).
- Implémenter un système de feedback utilisateur (évaluations des commerces, commentaires).

2. Approfondir l'aspect e-commerce

- Intégrer un panier pour des commandes locales (click & collect).
- Ajouter des passerelles de paiement simples (comme Stripe ou PayPal Sandbox pour l'entraînement).
- Permettre aux commerçants de gérer leurs stocks directement sur la plateforme.

3. Gestion de communauté locale

- Intégrer une section forum ou "mur d'annonces" pour les habitants (partage d'idées, recherches de services).
- Mettre en place un système de recommandations basé sur les préférences des utilisateurs (produits ou événements suggérés).

4. Ajout de gamification pour encourager l'engagement

- Offrir des badges ou points aux utilisateurs qui visitent plusieurs commerces ou participent à des événements.
- Créer un tableau des "ambassadeurs locaux" pour ceux qui partagent ou invitent des amis à utiliser la plateforme.

5. Intégrer des outils modernes

- Utiliser **Firebase** pour une gestion simplifiée des notifications en temps réel et de l'authentification.
- Ajouter une **Progressive Web App (PWA)** pour permettre une utilisation mobile hors connexion.
- Explorer les fonctionnalités de **GPT ou IA légère** pour une assistance utilisateur basique (FAQ ou suggestions personnalisées).

Détails supplémentaires :

1. Modules spécifiques pour les commerçants

- **Statistiques en temps réel :** Nombre de visites, taux de clics sur leurs produits ou événements.
- **Gestion avancée des offres :** Planification automatique des promotions (ex. : "Black Friday", Noël).

2. Accessibilité et inclusion

- S'assurer que le site respecte les normes d'accessibilité (ARIA).
- Ajouter des options de langue (par exemple, français, anglais, ou langues locales selon la région).

3. Déploiement et maintenance

- Mettre en place une stratégie DevOps simple : automatisation du déploiement avec Git Actions ou un CI/CD équivalent.
- Utiliser Docker pour standardiser l'environnement de développement et simplifier le déploiement.

Scénarios avancés pour rendre le projet réaliste

Scénario A : Support pour des artisans nomades

Certains artisans n'ont pas de boutique fixe (ex. : food trucks, artisans présents sur des marchés). La plateforme doit permettre :

- Une localisation dynamique (position GPS pour leurs horaires ou jours de présence).
- Une carte interactive pour suivre leur localisation en temps réel.

Scénario B : Gestion d'une communauté fermée

L'outil est destiné à une petite communauté, comme un groupe d'artisans ou commerçants d'une zone rurale :

- Création d'un espace privé réservé aux membres (extranet).
- Fonctionnalité de partage d'informations internes (tutoriels, documents).

Scénario C : Intégration écologique et durable

Pour sensibiliser les utilisateurs, intégrer des indicateurs écoresponsables :

- Afficher l'empreinte carbone approximative des livraisons locales.
- Mettre en avant des commerces avec des labels écologiques.

Ajouts pour le suivi pédagogique

- **Livrables intermédiaires :** Ajouter des jalons hebdomadaires pour le suivi (par exemple : validation des maquettes, livraison d'un MVP).
- **Ateliers intégrés :** Planifier des mini-ateliers pendant le projet pour approfondir des sujets spécifiques (ex. : bonnes pratiques Git, gestion de formulaire sécurisé).

1. Intégration de fonctionnalités spécifiques

- **Notifications temps réel :** Par exemple, envoyer des alertes en temps réel aux utilisateurs avec **Firebase Cloud Messaging**.
- **E-commerce simplifié :** Ajouter un panier d'achat et intégrer une API de paiement comme **Stripe** ou **PayPal**.
- **Cartes interactives :** Une intégration clé en main avec **Google Maps API** ou **Leaflet**, incluant géolocalisation et itinéraires.

2. Sécurisation du projet

- **Proposition :** Guide pratique pour gérer l'authentification et la protection des données utilisateurs.
- **Exemple concret :** Mise en œuvre de tokens JWT (JSON Web Token) ou sessions sécurisées avec hashing bcrypt.

3. Gestion de projet et organisation pédagogique

- **Proposition :** Structurer le projet en suivant une méthode agile (Scrum ou Kanban) adaptée au contexte de formation.
- **Exemple concret :** Création d'un tableau Trello ou Notion prêt à l'emploi pour suivre les étapes de développement et les livrables.

4. Accessibilité et performances

- **Proposition :** Comment optimiser le projet pour une utilisation rapide et accessible (normes WCAG + outils comme Lighthouse).
- **Exemple concret :** Réduire le temps de chargement en intégrant des images WebP et un système de mise en cache.

5. Déploiement du projet

Avant de déployer, quelques étapes importantes à vérifier :

5.1. Nettoyage du code

- Assure-toi que le code est bien organisé et sans erreurs (tests unitaires ou manuels).
- Supprime les fichiers inutiles ou sensibles (fichiers. env, clés API dans le code, etc.).

5.2. Créer un fichier. env

- Centralise les variables sensibles (clés API, URLs, etc.) dans un fichier.env.
- Utilise une librairie comme dotenv (Node.js) ou symfony/dotenv (PHP) pour charger ces variables.

5.3. Construire les fichiers pour production

- Si tu utilises un Framework front-end (React, Vue.js, Angular), exécute la commande de build :

  Bash

  ```
  npm run build
  ```

  Cela génère des fichiers optimisés dans un dossier (dist ou build).

5.4. Configurer le serveur back-end

- Ajoute un script pour démarrer l'application en production. Exemple avec Node.js :

  Json

```
"scripts": {
  "start": "node server.js"
}
```

Conseils pour un déploiement réussi

- **Tests rigoureux :** Avant le déploiement, teste le projet localement en mode production (npm run start).
- **Sauvegardes :** Conserve une copie locale ou sur Git pour chaque version déployée.
- **Surveillance :** Utilise des outils comme **New Relic** ou **Sentry** pour surveiller les performances et détecter les bugs.