Mémo Technique – Projet de Fin d'Année en Intelligence Artificielle

1. Contexte et Objectifs du Projet

Ce projet avait pour objectif de concevoir une application web interactive permettant à un utilisateur de transformer visuellement une pièce (salon, cuisine, etc.) en fonction d'une époque (par exemple : années 1970, contemporain) et d'un style esthétique (cyberpunk, rustique, industriel, etc.).

L'utilisateur sélectionne une pièce et personnalise son apparence via une interface React. Une requête est ensuite envoyée à une API d'intelligence artificielle (modèle Hugging Face) pour générer une image correspondant à ces critères. L'ambition pédagogique était de croiser design d'interface, logique de programmation et exploitation d'un modèle génératif d'images.

---

2. Technologies et Architecture

- **Frontend** :
    - React (avec Vite pour le bundling)
    - TailwindCSS pour le style
    - Icônes via la bibliothèque `lucide-react`
- **Backend / Génération IA** :
    - API de Hugging Face
    - Modèle utilisé : `lllyasviel/control_v11p_sd15_canny` (génération d'image guidée par détection de contours)
- **Autres outils** :
    - npm (v10.9.2)
    - Node.js (v22.16.0)

- Vite (v6.3.5)
- Explorateurs d’images publiques pour les tests (placekitten, Unsplash…)

---

3. Fonctionnalités Réalisées

- Interface React opérationnelle
- Plan interactif avec pièces cliquables
- Sélection dynamique de l’époque et du style
- Génération automatisée d’un prompt à partir des choix utilisateur
- Simulation de génération d’image via `fetch` d’URL d’images
- Affichage de l’image générée
- Galerie des dernières images créées
- Bouton de téléchargement pour enregistrer l’image finale

---

4. Problèmes Techniques Rencontrés

A. Écran vide après le clic sur une pièce

- **Cause** : la fonction `generatePrompt()` était appelée avant d’être définie, entraînant un crash silencieux de React.
- **Solution** : ajouter une vérification sur la sélection de la pièce (`selectedRoom`) avant d’appeler la fonction.

B. Appels API Hugging Face échoués

- **Cause** : les images d’entrée étaient hébergées sur Unsplash, un service protégé par CORS (Cross-Origin Resource Sharing), rendant les requêtes impossibles.

- **Solution** : utiliser des images locales ou des services non-protégés comme `placekitten.com`.

C. Erreurs réseau ou écran blanc inexpliqué

- **Cause** : absence de logs dans les fonctions critiques, rendant le débogage difficile.
- **Solution** : ajout systématique de `console.log()` dans les fonctions `generateImage()` et dans les composants.

D. Prompt incorrect ou non généré

- **Cause** : des champs comme `selectedEpoch` ou `selectedStyle` étaient parfois vides.
- **Solution** : désactiver le bouton "Transformer" tant que les champs requis ne sont pas remplis.

E. Le bouton "Transformer" ne déclenchait aucune action

- **Cause** : absence d'événement `onClick`, ou problème dans la gestion des états (`useState`).
- **Solution** : ajout de logs, vérification des états, et création d'une fonction `resetSelection()` pour éviter les conflits.

---

5. Leçons Apprises

- Toujours encapsuler les fonctions dépendantes de l'état dans des conditions de sécurité (`if (selectedRoom)...`).
- Utiliser systématiquement la console de développement (F12) pour repérer erreurs ou blocages invisibles.
- Tester les appels d'API avec des ressources sûres (fichiers locaux ou services sans CORS).
- Ajouter des `console.log()` à toutes les étapes importantes de génération ou de transition d'état.

- Anticiper une meilleure gestion des erreurs côté utilisateur (alertes, messages d’erreur, visuels de secours).

- Prendre le temps de bien structurer les composants React dès le début pour éviter les réécritures inutiles.

---

6. Conseils pour les Futurs Étudiants

- Commencez l’intégration de l’intelligence artificielle le plus tôt possible. Même une version très simple est précieuse pour les tests.

- Utilisez en priorité des images locales ou hébergées sur des serveurs qui n’imposent pas de restrictions CORS.

- Développez dès le départ une fonction solide de génération de prompt, et testez-la indépendamment du reste.

- Utilisez un système de gestion de version comme GitHub dès les premières lignes de code.

- Restez minimalistes dans la conception de l’interface utilisateur avant d’ajouter des effets visuels ou des animations.

- Prenez en compte les contraintes de version des outils (Node.js, npm, Vite...) dès l’installation du projet pour éviter les bugs de compatibilité.

---

7. Conclusion

Le projet a atteint un état fonctionnel satisfaisant en mode simulation, avec une interface interactive et une logique complète de transformation. Cependant, la connexion finale avec l’API Hugging Face pour la génération réelle d’images a rencontré des blocages techniques importants (erreurs CORS, crashs silencieux, gestion incomplète des états).

Malgré cela, le travail accompli constitue une base solide et pédagogique pour tout étudiant souhaitant poursuivre ou améliorer un projet similaire. Ce mémo récapitule les principales erreurs rencontrées, les solutions apportées, ainsi que les meilleures pratiques pour éviter que d'autres étudiants ne perdent un temps précieux sur les mêmes points bloquants.