Rapport de la Phase de Conception du Projet StageAI

1. Présentation Générale

Le projet StageAI est une plateforme web qui combine plusieurs technologies modernes pour offrir de
- backend/ : API et logique métier (Flask, IA, MongoDB)
- client/ : Application React pour la gestion de librairie
- frontend/ : Application Next.js pour l'interface IA moderne

2. Architecture Générale

a. Architecture en couches

Frontend (Next.js) : Interface utilisateur moderne pour l'IA (génération de texte, d'images, chat).
Client (React) : Application de gestion de librairie (livres, utilisateurs, commandes, etc.).
Backend (Flask) : API REST, gestion de l'authentification, génération IA, stockage MongoDB.

b. Communication

Les interfaces frontend et client communiquent avec le backend via des requêtes HTTP (fetch/AJAX).
Le backend expose des endpoints REST pour chaque fonctionnalité (auth, génération, historique, etc.

3. Conception du Backend

a. Technologies

Flask : Framework web Python léger pour l'API.
MongoDB : Base de données NoSQL pour stocker utilisateurs et historiques.
Stable Diffusion (diffusers) : Génération d'images IA.
OpenAI API : Génération de texte/chat IA.
Flask-Bcrypt : Sécurisation des mots de passe.
Flask-CORS : Autorisation des requêtes cross-origin.

b. Structure des endpoints

/register, /login, /logout : Authentification utilisateur.
/generate-text : Génération de texte IA (avec historique).
/generate-image : Génération d'image IA (avec historique).
/history : Récupération de l'historique utilisateur.
/chat : Chat IA (GPT-3.5).
/images/<filename> : Accès aux images générées.

c. Sécurité

Utilisation de sessions Flask pour l’authentification.
Hashage des mots de passe avec Bcrypt.
Gestion des erreurs et des statuts HTTP.

4. Conception du Frontend (Next.js)

a. Technologies

Next.js : Framework React pour SSR/SSG, pages dynamiques.
TypeScript : Typage statique.
Tailwind CSS : Design moderne et responsive.
Framer Motion : Animations fluides.

b. Fonctionnalités

Génération de texte et d’images via prompts utilisateur.
Chat IA interactif.
Affichage dynamique des résultats et de l’historique.
Appels API vers le backend Flask.

c. Structure

app/page.tsx : Page principale avec UI pour toutes les fonctionnalités IA.
app/layout.tsx : Layout global, gestion des polices et du thème.
public/ : Fichiers statiques (icônes, images).
globals.css : Styles globaux (avec Tailwind).

5. Conception du Client (React)

a. Technologies

React : SPA pour la gestion de librairie.
React Router : Navigation entre pages.
Bootstrap : UI classique et responsive.

b. Fonctionnalités

Gestion des livres, utilisateurs, commandes, emprunts.
Authentification et profils.
Interface d’administration (pages dédiées pour chaque ressource).
Appels API (ex : /api/books).

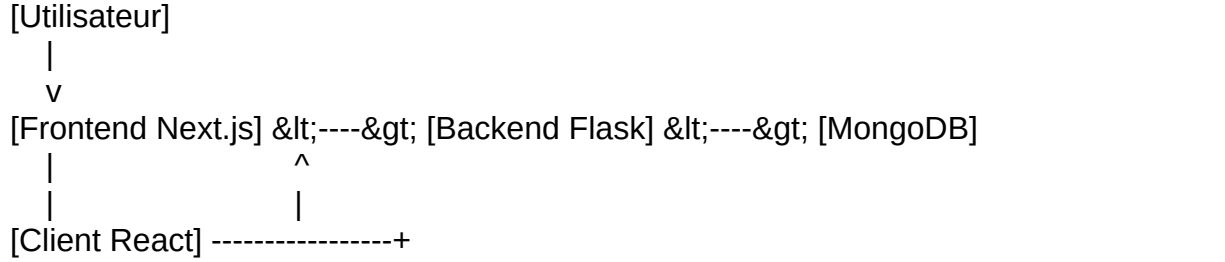
c. Structure

src/pages/ : Pages principales (Home, Books, Cart, Admin, etc.).
src/components/ : Composants réutilisables (Navbar, Footer).
AppRouter.js : Définition des routes de l’application.

6. Modélisation des Données

Utilisateurs : username, password (hashé)
Historique : username, type (texte/image), prompt, résultat, date
Livres (dans client) : title, author, etc. (dépend du backend non fourni ici)

7. Diagramme de Conception (exemple textuel)



- Les deux interfaces (Next.js et React) peuvent communiquer avec le backend Flask.
- Le backend centralise la logique métier et l’accès aux données.

8. Choix de conception

Séparation claire des responsabilités : chaque dossier a un rôle précis.
Scalabilité : possibilité d’ajouter d’autres modules IA ou de gestion.
Sécurité : gestion des sessions, hashage des mots de passe.
Expérience utilisateur : UI moderne (Next.js) et classique (React/Bootstrap).

9. Conclusion

La phase de conception a permis de structurer le projet de façon modulaire, sécurisée et évolutive, en