Cahier de Charge Fonctionnelle

Projet : LabGenius : Simulateur de Laboratoire de Synthèse Génétique

Niveau	Mage de rang A
Formateur	**Mr Webster Sturgis MITHOU MBENGA** Hacker éthique, Game codeur & Ingénieur en génie logiciel

CONTEXTE ET OBJECTIFS DU PROJET

A) CONTEXTE

Vous êtes une équipe de développement mandatée par une jeune startup biotechnologique, "**GenoSoft**", pour créer la preuve de concept (PoC) de leur prochaine application web. Cette application, nommée LabGenius, a pour but de simuler le travail quotidien d'un chercheur dans un laboratoire de synthèse génétique. L'objectif est de créer un outil pédagogique et de démonstration pour aider les étudiants et les nouveaux investisseurs à comprendre le processus complexe de manipulation génétique.

1. Objectifs Pédagogiques et Techniques

- Mise en pratique du développement front-end : Maîtriser HTML pour la structure, CSS pour le design (responsive, moderne) et JavaScript pour l'interactivité et la logique métier.
- Conception d'une application monopage (SPA) ou multi-pages cohérente : Créer une expérience utilisateur fluide avec une navigation claire entre les différentes sections du laboratoire.
- Modélisation et simulation de données : Manipuler des données complexes (séquences d'ADN, enzymes, protocoles) en PHP (objets, tableaux) et simuler des processus biologiques (transcription, mutation).
- Travail en équipe et gestion de projet : Se répartir les tâches, utiliser un système de versionnement (Git) et présenter un produit fonctionnel.
- CULTIVER LA QUALITÉ LOGICIELLE : L'objectif sous-jacent mais fondamental est d'apprendre à écrire un code propre, maintenable, robuste et professionnel, même pour une application simple.

2. Description fonctionnelle de l'application

L'application "LabGenius" doit simuler un laboratoire virtuel. L'utilisateur (le chercheur) peut interagir avec différents "équipements" et "stations de travail" pour réaliser des expériences de synthèse génétique.

2.1. Pages / Sections de l'Application

1. Page d'Accueil / Tableau de Bord (Dashboard) :

- Présentation du laboratoire virtuel.

- Afficher un résumé des projets en cours, des séquences d'ADN récemment synthétisées ou des journaux de laboratoire (logs).
- Intégrer des indicateurs visuels simples (ex: nombre de séquences stockées, statut des expériences).

2. Page "Séquenceur" / Éditeur de Génome :

- Fonctionnalité clé : Permettre à l'utilisateur de visualiser et d'éditer une séquence d'ADN.
- Affichage d'une séquence d'ADN exemple (ex: "ATG-CGT-AAA-TGC") sous forme de chaîne de caractères ou de manière plus graphique (blocs de couleurs pour les bases A, T, G, C).
- Boutons pour simuler des mutations ponctuelles (substitution d'une base) ou l'insertion d'un gène.
- Un historique simple des modifications effectuées.

3. Page "Machine de Synthèse" :

- Simuler le processus de synthèse d'une séquence d'ADN.
- L'utilisateur saisit une séquence d'ADN (ou la charge depuis l'éditeur) et lance la "synthèse".
- Une barre de progression animée simule le temps de synthèse.
- À la fin, afficher un rapport de succès/échec (avec un taux de réussite basé sur la complexité de la séquence - logique aléatoire ou déterministe à définir par les étudiants).

4. Page "Bibliothèque Génomique" :

- Afficher une liste de séquences d'ADN pré-enregistrées (ex: "Gène de la fluorescence", "Séquence de résistance à un antibiotique").
- Possibilité de "charger" une séquence de la bibliothèque vers l'éditeur de génome (en utilisant le `localStorage` par exemple).
- Ajouter la possibilité de "marquer comme favorite" une séquence.

2.2. Fonctionnalités Attendues (Back-end simulé en PHP)

- Gestion de l'état de l'application : Utiliser des variables et des objets php pour stocker les données (séquences, protocoles, historique).
- Manipulation les données : Créer, modifier et supprimer dynamiquement des éléments HTML en fonction des actions de l'utilisateur (afficher une nouvelle séquence, ...).

- Gestion d'événements : Réagir aux clics sur les boutons, aux soumissions de formulaires (même simples) et aux changements de valeurs dans les champs de saisie.
- Simulation de processus :
- Mutation : Implémenter la logique de modification aléatoire d'une lettre dans une séquence d'ADN.
- Persistance de données simple : Utiliser les fonctions adequates pour sauvegarder les séquences favorites ou les dernières séquences synthétisées.
- Interface responsive : L'application doit être utilisable sur ordinateur et sur tablette non sur un smartphone.

3. Exigences de Qualité Logicielle

Au-delà des fonctionnalités, la qualité de votre code et de votre démarche sera évaluée rigoureusement. Voici les bonnes pratiques à appliquer impérativement :

3.1. Architecture et Organisation du Code

- Séparation des préoccupations : Séparez clairement :
  - HTML : La structure et le contenu.
  - CSS : La présentation et le style.
  - JavaScript : Le comportement.
  - PHP: la logique

- Modularité : Divisez votre code PHP en différents fichiers ou modules logiques si le projet devient conséquent
- Principe DRY (Don't Repeat Yourself) : Identifiez les actions répétitives et factorisez-les dans des fonctions réutilisables.

3.2. Qualité et Lisibilité du Code

- Nommage explicite : Utilisez des noms de variables, fonctions et classes clairs et auto-documentés (en anglais ou en français, mais de manière cohérente). Ex: `displayGeneSequence` au lieu de `afficherDonnees`.
- Indentation et formatage : Le code doit être correctement indenté et formaté de manière cohérente.

- Commentaires pertinents : Commentez le "pourquoi" pas le "quoi". Le code doit expliquer le "comment". Utilisez des commentaires pour expliquer les algorithmes complexes ou les choix de conception.
- Constantes : Utilisez des constantes (convention `UPPER_SNAKE_CASE`) pour les valeurs qui ne changent pas (ex: `const VALID_BASES = ['A', 'T', 'G', 'C'];`).

3.3. Robustesse et Gestion d'Erreurs

- **Validation des entrées utilisateur :** Ne faites jamais confiance à l'utilisateur. Vérifiez que les séquences d'ADN saisies ne contiennent que des caractères valides (A, T, G, C) avant de les traiter. Affichez des messages d'erreur clairs.
- **Gestion des cas limites :** Que se passe-t-il si la séquence est vide ? Si l'utilisateur clique sur "muter" alors qu'aucune séquence n'est chargée ? Si `localStorage` est vide ?
- **Blocs `try...catch`** : Utilisez-les pour les opérations qui peuvent échouer

3.4. Performance et Expérience Utilisateur

- Manipulation efficace du DOM : Évitez de manipuler le DOM en boucle. Construisez le HTML dynamique dans une chaîne de caractères et injectez-le en une seule fois.
- Feedback utilisateur : Toute action (lancer une synthèse, enregistrer, charger) doit avoir un retour visuel ou textuel immédiat. Désactivez les boutons pendant les opérations longues (ex: pendant la synthèse) pour éviter les clics multiples.

3.5. Travail en Équipe et Versionnement (Git)

- **Commits atomiques et fréquents :** Chaque commit doit représenter une modification logique et cohérente (ex: "Ajout de la fonction de validation ADN", "Correction du style de la page Bibliothèque"). Les messages de commit doivent être explicites (en anglais ou français).
- **Branches :** Utilisez des branches pour développer chaque fonctionnalité majeure (`feature/sequenceur`, `feature/bibliotheque`) et fusionnez-les via des pull requests

4. Contraintes Techniques et Livrables

4.1. Stack Technologique

- **Front-end :** HTML5, CSS3, JavaScript, (aucun framework comme React, Vue ou Angular n'est autorisé pour ce projet).
- Back – end : PHP
- Versionnement : Git, avec un dépôt sur GitHub ou GitLab (le lien devra être fourni). L'historique des commits doit refléter le travail de chaque membre.
- Design : Libre, mais doit être soigné, cohérent et évoquer un univers scientifique/technologique. L'utilisation de CSS Grid ou Flexbox est fortement recommandée pour la mise en page.

4.2. Livrables Attendus

1. Le code source complet de l'application, déposé sur un repository GitHub public.
2. Un fichier `README.md` à la racine du projet contenant :
   - Le titre du projet et les noms des membres de l'équipe.
   - Une description du projet.
   - Les instructions pour lancer et utiliser l'application (simple ouverture du `index.html` dans un navigateur).
   - Une courte liste des fonctionnalités implémentées.
   - Une section "Choix de conception et qualité logicielle" expliquant brièvement comment vous avez appliqué les bonnes pratiques (architecture, DRY, validation...).
   - En fichier pdf, une explication de l'architecture du code et des choix de conception.
   - Une démonstration de la qualité du code (montrer des exemples de fonctions bien écrites, de validation d'entrées, de gestion des erreurs).
   - Un retour sur les difficultés rencontrées et les solutions trouvées.

5. Évaluation et Critères de Réussite

Le projet sera évalué selon les critères suivants, avec un poids important accordé à la qualité :

- **Fonctionnalités (30%) :** L'application remplit-elle les fonctions décrites ? La simulation est-elle crédible et interactive ?
- **QUALITÉ LOGICIELLE (40%) : CRITÈRE MAJEUR**.

- **Architecture et modularité (15%) :** Le code est-il bien organisé, modulaire et respecte-t-il la séparation des préoccupations ?
- **Robustesse (15%) :** Les entrées sont-elles validées ? Les erreurs sont-elles gérées ? L'application plante-t-elle facilement ?
- **Lisibilité et conventions (10%) :** Le nommage est-il clair ? Le code est-il commenté de manière pertinente et bien formaté ?
- **Interface utilisateur et expérience (15%) :** L'application est-elle agréable à utiliser, intuitive et responsive ? Le design est-il cohérent avec le thème ?
- **Utilisation de Git et travail d'équipe (15%) :** Le repository est-il bien organisé ? Les commits sont-ils fréquents, atomiques et explicites ? La stratégie de branches a-t-elle été utilisée ?

6. Suggestions d'Amélioration (Options Bonus sur session)

Pour les groupes les plus avancés ou qui souhaitent aller plus loin :

- **Visualisation graphique :** Au lieu d'une simple chaîne de caractères, représenter la séquence d'ADN avec des cercles colorés
- **Ajout d'un "Carnet de laboratoire" :** Une page supplémentaire avec des notes que l'utilisateur peut ajouter/modifier et qui sont sauvegardées.
- **Thème sombre/clair :** Ajouter un bouton pour basculer entre un thème clair et un thème sombre.