1️⃣ Partie en anglais

I will start by presenting the architecture of my project.
At first, it seems complicated, but it is designed to keep the code clear, organize responsibilities, and put each feature in the right layer.

The project is divided into four main parts:

Models => store the data and logic rules, like user information or password handling. They define what is stored and how it's managed.

Controllers => contain the main logic of the application. They coordinate actions such as registration, login, and profile management.

Viewers =>handle the user interface and interactions. They display the data prepared by the controllers and update the interface dynamically.

Finally, the **HTML pages** define the structure and give access to the different parts of the application.
I will now briefly show some key files to illustrate this architecture, and then I'll continue the rest of the presentation in French.

2️⃣ Partie en français – développement et choix techniques – 10 à 15 min

A. Maquette et conception

Au début, j'ai eu beaucoup de mal avec la maquette et la prise en main de **Figma**, ça m'a pris pas mal de temps.
Pour ne pas perdre de temps, j'ai commencé à coder **le système de connexion et d'inscription** avant d'avoir terminé la maquette complète.

B. Identité visuelle

L'idée du design et de la charte graphique s'est construite progressivement.
J'avais une **image existante** provenant d'un ancien projet ou d'une idée non exploitée. Je l'ai utilisée comme **background pour le header**, et à partir de cette image j'ai défini **la palette de couleurs et le style visuel général** du projet.

C. Pages initiales et concept

Une fois le système de connexion terminé, je me suis retrouvée avec **une page d'accueil et un profil vide**. Il était difficile d'avancer sans une vision claire du projet.

L'idée de **CelestIA** est née naturellement : un site centré sur l'**intelligence artificielle**, un outil que j'utilise énormément, et qui est très actuel.

D. Développement fonctionnel

- Mise en place des **IA sur la page d'accueil**.
- Création d'un **système d'achat simulé**, pour récupérer les IA dans le profil et rendre le tout crédible.
- Importation des **données dans le profil**, comme les IA et les informations de connexion, via **localStorage**. J'ai simulé la persistance de session avec `isLogged` plutôt que d'utiliser `sessionStorage` directement — un petit raccourci pratique, qu'un dev « pas très patient » pourrait faire… même si en vrai, ce n'est pas un problème.

E. Architecture et gestion du code (version enrichie et honnête)

Pendant le projet, j'ai revu plusieurs fois l'architecture et la façon de coder.
Au départ, j'utilisais `<script defer>`, ce qui fonctionne pour exécuter le JS après le chargement du DOM, mais tout le code était global et difficile à organiser.
Quand j'ai commencé à gérer le header dynamique, je me suis un peu perdue et j'ai introduit les modules ES6 avec `type="module"` et import/export.
Un module ES6, c'est un fichier JS avec son scope propre, qui permet d'exporter et d'importer uniquement ce dont on a besoin entre fichiers.
J'ai mis le header dans un module pour pouvoir réutiliser le code facilement, garder un scope propre et faciliter l'évolution de l'application.
Avec du recul, j'aurais peut-être voulu mieux séparer les parties de l'interface en components, comme le header ou le footer, pour que chaque partie soit complètement autonome et réutilisable, mais c'est une piste d'amélioration pour la suite.

F. Sécurité minimale

- Même en utilisant localStorage, il fallait **hasher les mots de passe**.
- J'ai implémenté un **hashage basique** sans importer de bibliothèques externes. Pour un vrai projet, je sécuriserais mieux, mais ici c'était suffisant.

G. Amélioration de l'expérience utilisateur (UI/UX)

- Suppression des alertes brutes.
- Mise en place de **messages pop-up** pour rendre l'expérience utilisateur plus fluide et agréable