UNIVERSITÉ PARIS NANTERRE

Licence MIASHS deuxième année

Rapport de projet informatique

Le titre du rapport de stage

Projet réalisé en 2025

Projet Secret Hitler

Membres du groupe

Taha Turkan – 44009891
Anasse Bouydarne – 44014032

Dépôt GitHub :
`https://github.com/Captain78lii/Projet-Secret-Hitler`

Table des matières

1 Introduction

1.1 Contexte

Ce projet consiste à développer une version web du jeu de société *Secret Hitler* à l'aide des technologies HTML, CSS et JavaScript. Le jeu repose sur des mécanismes de stratégie et de déduction, offrant un cadre pertinent pour représenter les interactions entre joueurs sous forme de graphes. Le projet s'inscrit également dans une démarche de développement assisté par l'intelligence artificielle.

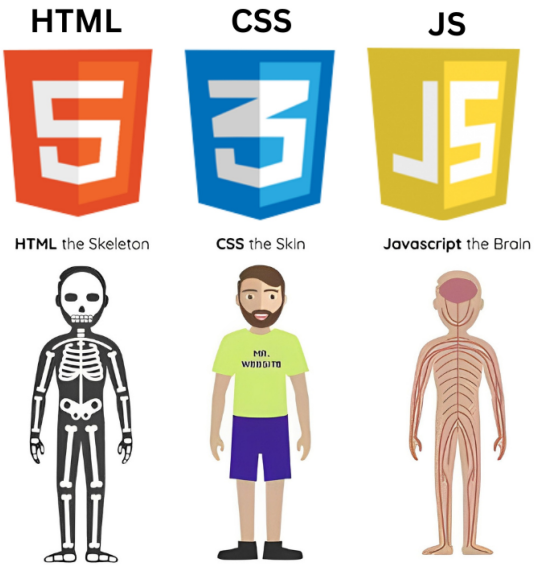
1.2 Problématique et objectifs

L'objectif principal du projet n'est pas uniquement de réaliser un jeu parfaitement abouti, mais de mettre en œuvre une approche de développement assistée par l'IA. Le projet vise à explorer l'utilisation d'un LLM comme outil d'aide à la conception, à la résolution de problèmes et à l'amélioration progressive du code. Les objectifs sont donc :

- concevoir un jeu web fonctionnel reproduisant les règles de *Secret Hitler* ;
- utiliser l'IA pour discuter des choix techniques et corriger les erreurs rencontrées ;
- représenter les interactions entre joueurs à l'aide de graphes dynamiques ;
- améliorer progressivement le projet grâce aux échanges avec l'IA.

2 Environnement de travail

2.1 Langages et technologies



2.2 Outils de développement

— **Éditeur de code :**

— **Navigateurs :**

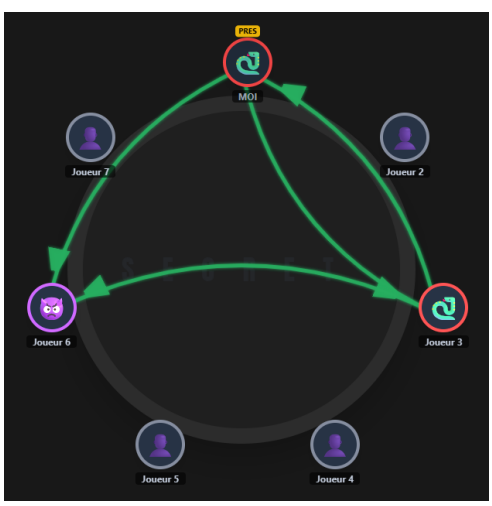
— **Gestion de versions :**

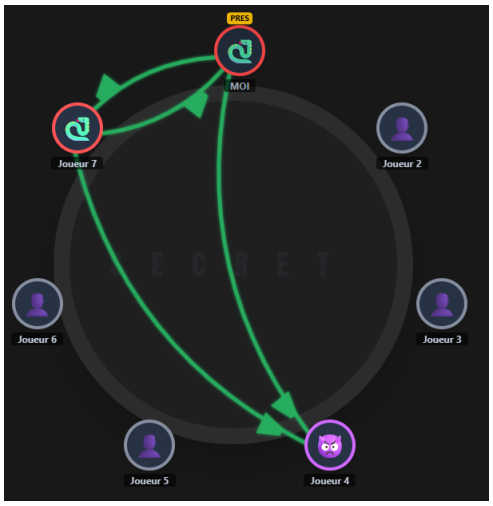
GitHub

2.3 Graphes et visualisation

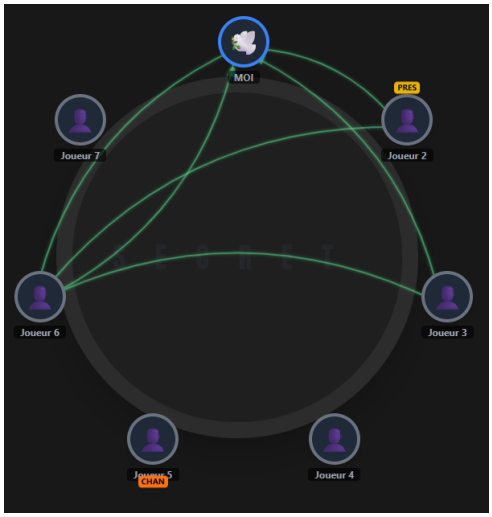
Dans le cadre du projet, les graphes sont utilisés pour représenter les interactions entre les joueurs au cours de la partie.

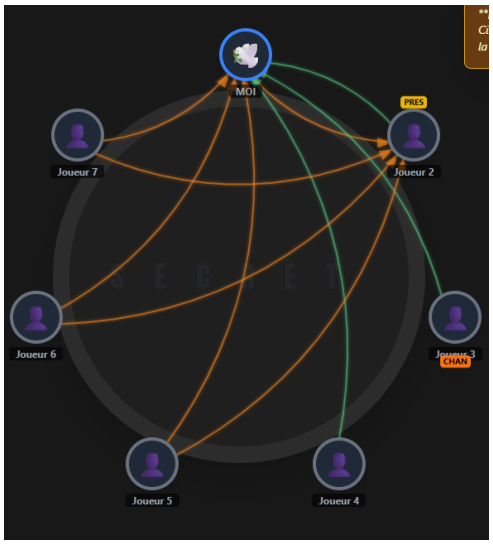
Nœuds (sommets) : Chaque joueur du jeu est représenté par un nœud du graphe. Les nœuds peuvent contenir des informations comme : le rôle supposé (libéral / fasciste / Hitler), le niveau de suspicion, l'historique des votes.





— Ces flèches montrent seulement vos alliés fascistes (si vous êtes libéral aucune de ces flèches apparaîtrons).

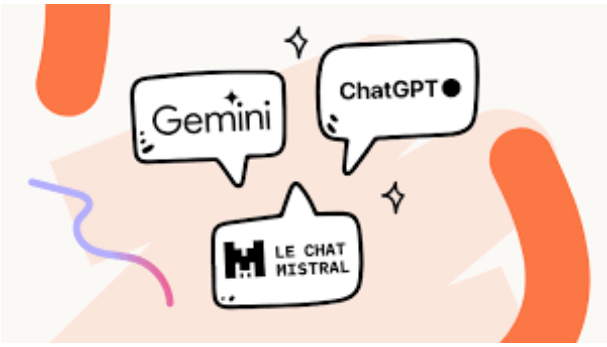




— **Arêtes (liens) :** Une arête entre deux nœuds représente une interaction entre deux joueurs. Exemple : la confiance en vert, la suspicion en orange. Ces arêtes évoluent au cours de la partie selon les actions des joueurs.

2.4 Utilisation de l'intelligence artificielle

Le projet étant assisté par l'IA, un modèle de type *Large Language Model* (LLM) a été utilisé pour accompagner le développement.



L'IA a permis :

— d'aider à la compréhension et à la conception de certaines parties du code,
— de proposer des solutions aux problèmes rencontrés,
— d'améliorer la structure et la lisibilité du code,
— d'accompagner la réflexion tout au long du projet. //

L'IA peut analyser le graphe pour :

— détecter des comportements suspects,
— proposer des décisions,
— aider à simuler des joueurs artificiels.

3 Description du projet et objectifs

3.1 Fonctionnement général du jeu

Le projet consiste à développer une application web reproduisant le jeu de société *Secret Hitler*. Le jeu se déroule en plusieurs tours durant lesquels les joueurs doivent voter, prendre des décisions collectives et tenter d'identifier les rôles secrets. L'application gère automatiquement les différentes phases du jeu, les votes, ainsi que l'évolution de la partie en respectant les règles officielles.

3.2 Objectifs pédagogiques et techniques

Ce projet a pour objectif de mettre en pratique les connaissances acquises en programmation web et en théorie des graphes. Les objectifs sont à la fois pédagogiques et techniques :

- développer une application web interactive en HTML, CSS et JavaScript ;
- modéliser les relations entre joueurs à l'aide de graphes dynamiques ;
- utiliser l'intelligence artificielle pour assister le développement du projet ;
- travailler en équipe à l'aide d'outils collaboratifs comme GitHub.

4 Difficultés rencontrées et Solutions Techniques

Dans le cadre du développement de l'adaptation web de *Secret Hitler*, nous avons rencontré plusieurs défis techniques et conceptuels. Voici le détail de ces obstacles et des solutions apportées.

4.1 La Modélisation et la Visualisation du Graphe de Confiance

La Difficulté : Il fallait représenter graphiquement les relations entre les joueurs (qui soupçonne qui) via des courbes dynamiques. La difficulté résidait dans le calcul mathématique des relations et la gestion de la "visibilité sélective".

La Résolution : Nous avons défini un graphe orienté et pondéré. Chaque arête possède un poids entre -1.0 (Conflit) et +1.0 (Alliance) (Figure 1).

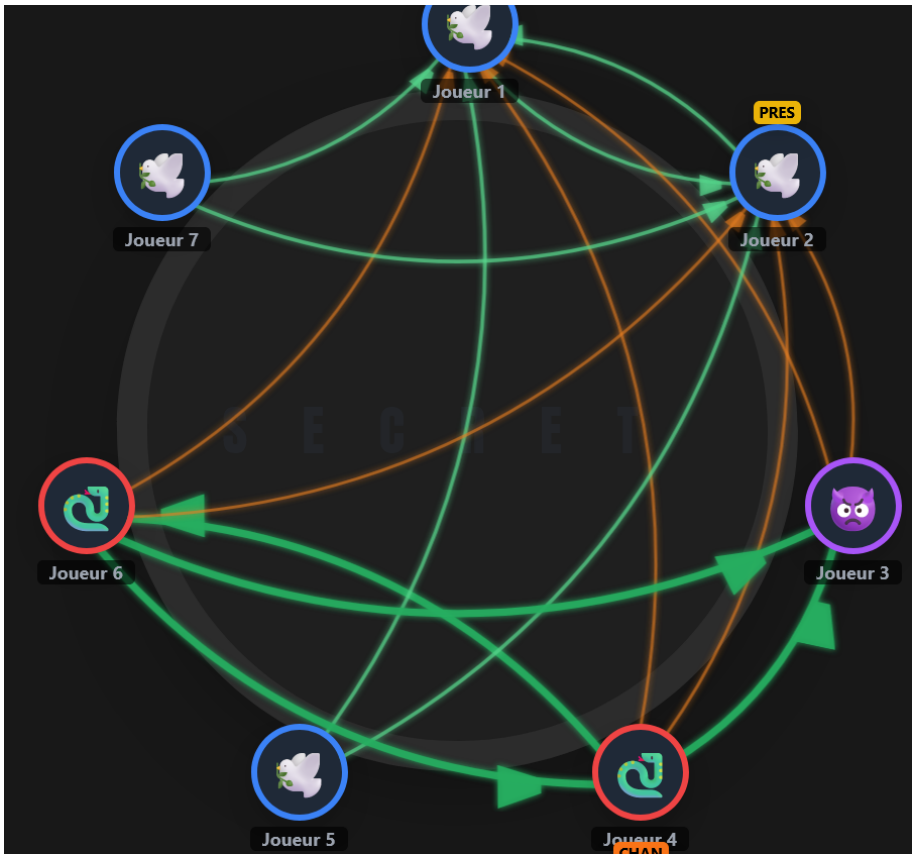


FIGURE 1 – Définition technique des arêtes pondérées

De plus, la logique du jeu influence directement ces couleurs (orange, vert) (Figure 2).

C'est compris. Je vais mettre à jour la logique du jeu pour que les actions (promulgation de lois) influencent la confiance des joueurs.

Désormais :

1. Si une **Loi Fasciste** passe : Les Libéraux vont fortement se méfier (liens **Orange**) du Président et du Chancelier.
2. Si une **Loi Libérale** passe : Les Fascistes vont se "méfier" (considérer comme opposants) du gouvernement.

Cela crée une dynamique visuelle où les alliances et les suspicions (lignes orange) se forment naturellement en fonction des actions de chaque gouvernement.

ons ou créons ensemble

FIGURE 2 – Règles d'évolution du graphe selon les lois votées

4.2 L'Intelligence Artificielle des Bots (Prise de Décision)

La Difficulté : Créer des bots qui ne jouent pas totalement au hasard, mais qui simulent un comportement humain cohérent (méfiance envers les

fascistes).

La Résolution : Création d'une **Matrice de Confiance** (`trustMatrix`). Le vote des bots est calculé selon la moyenne des confiances envers le Président et le Chancelier.

2. **L'Algorithme de Vote** : Au moment de voter pour un gouvernement, le bot ne tire pas au sort. Il exécute un calcul précis basé sur ses relations actuelles :
   - Il récupère son score de confiance envers le **Président** actuel.
   - Il récupère son score de confiance envers le **Chancelier** proposé.
   - Il fait la **moyenne** de ces deux scores.
   - Il ajoute un "biais" (`bias`) selon son propre rôle (un fasciste aura tendance à voter plus facilement "OUI" pour semer le chaos).
   - Si le résultat final est supérieur à un seuil défini (ex: `-0.2`), le bot vote **JA**. Sinon, il vote **NEIN**.

FIGURE 3 – Détail de l'algorithme de vote pondéré

4.3 Le Rythme du Jeu et la Narration (UX)

La Difficulté : Initialement, le code exécutait les tours instantanément. L'information défilait trop vite dans les logs pour être comprise par le joueur.

La Résolution : Développement d'un système de **"Queue d'Annonces"**. Le jeu se met en pause visuelle pour afficher les événements majeurs.

FIGURE 4 – Overlay d'annonce : Une loi Fasciste est promulguée

4.4 Gestion de l'Aléatoire et Mise en Scène

La Difficulté : Manque de sensation d'aléatoire lors de l'attribution des rôles (sentiment de répétition).

La Résolution : Correction de l'algorithme de mélange (*Fisher-Yates*) et création d'une animation "Roulette" pour matérialiser le hasard.

FIGURE 5 – Attribution Libéral

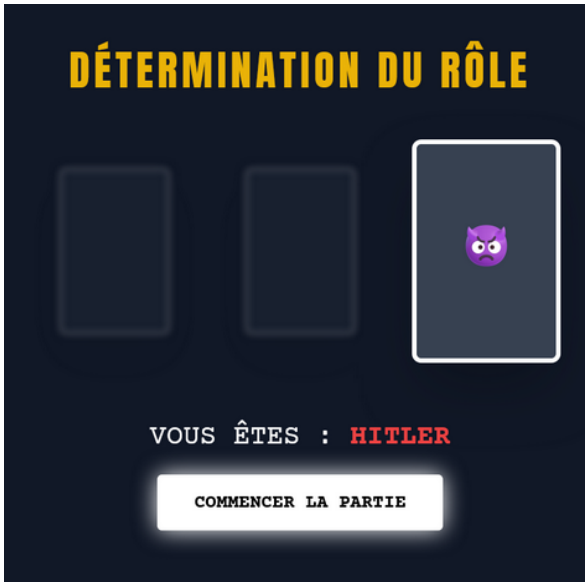


FIGURE 6 – Attribution Hitler

4.5 La Synchronisation des États et les Boucles Infinies

La Difficulté : Le jeu fonctionne comme une "machine à états" séquentielle, alors que React cherche à mettre à jour l'affichage instantanément. Nous avons rencontré des problèmes de **boucles infinies** (re-renders en chaîne) causées par des bots modifiant l'état trop rapidement.

La Résolution : Nous avons dû imposer un séquençage strict :

— Utilisation de délais (`setTimeout`) pour découpler la logique de jeu du rendu.

— Pauses obligatoires pour permettre aux animations (Flash Info) de se terminer.

Pour remédier à ce chaos technique, nous avons agi comme des agents de la circulation. Nous avons imposé un séquençage strict au code : au lieu de laisser l'application réagir instantanément à chaque modification (ce qui créait la boucle), nous avons utilisé des "délais" (setTimeout) et des conditions précises. Concrètement, nous avons forcé le jeu à faire des pauses obligatoires entre chaque étape logique. Cela permet à l'interface de finir son animation et de se stabiliser avant que le code ne lance l'action suivante, brisant ainsi le cycle infernal de redémarrage.

FIGURE 7 – Exemple d'événement asynchrone (Flash Info) géré sans bloquer le rendu

4.6 La Gestion de l'Audio et l'Autoplay

La Difficulté : Les navigateurs bloquent la lecture automatique (Autoplay Policy). De plus, les changements de page React coupaient la musique.

La Résolution : Création d'un écran "Start" obligeant l'interaction et mise en place d'un **Singleton Audio**.

FIGURE 8 – Écran d'accueil forçant l'interaction utilisateur pour activer l'audio

FIGURE 9 – Agrandissement du message d'avertissement à l'utilisateur

5 Bilan

Ce projet a permis de développer une application web inspirée du jeu de société *Secret Hitler*, en combinant programmation web, théorie des graphes et utilisation de l'intelligence artificielle. La modélisation des interactions entre joueurs sous forme de graphes a permis d'appliquer concrètement les notions vues en cours. Le travail en groupe et l'assistance par l'IA ont également contribué à la réussite du projet.

6 Annexes

6.1 Exemple d'exécution du projet

FIGURE 10 – Lancement du jeu

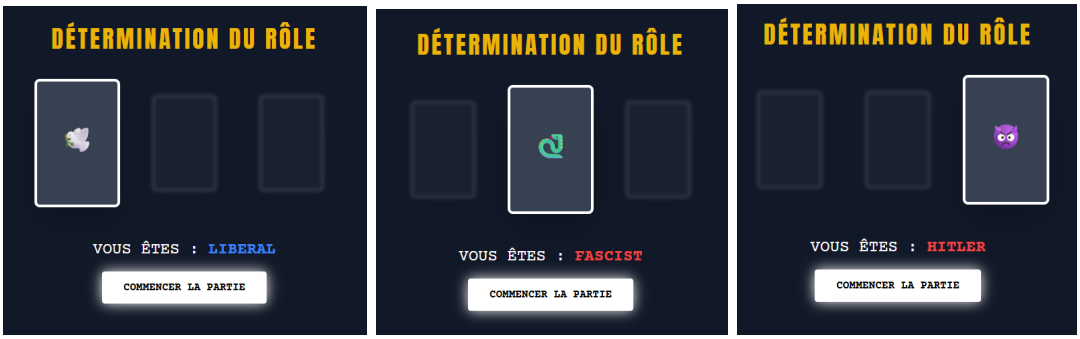


FIGURE 11 – Détermination du rôle

FIGURE 12 – Les Aides et Analyses

6.2 Manuel utilisateur

6.2.1 Rôles des joueurs

Chaque joueur reçoit secrètement un rôle au début de la partie :

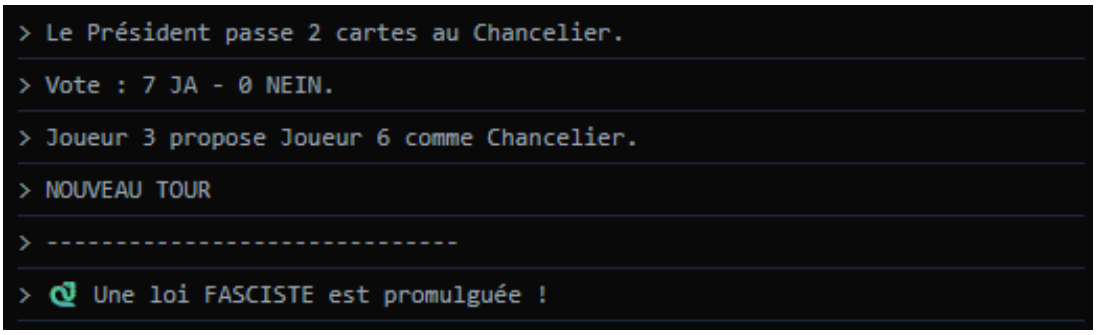


FIGURE 13 – Le déroulement du jeu

FIGURE 14 – La pioche et le décompte (30sec)

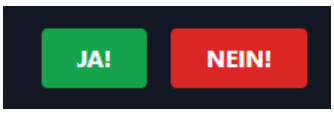
— Les **Libéraux** ne connaissent l'identité de personne.
— Les **Fascistes** connaissent les autres fascistes.
— **Hitler** ne connaît pas les fascistes (dans les parties à plus de six joueurs).

6.2.2 Déroulement d'un tour

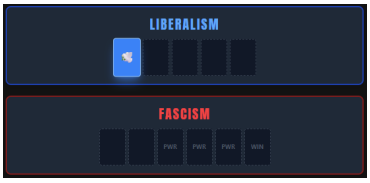
À chaque tour :
— Un Président est désigné.

*

— Le Président propose un Chancelier.



— Tous les joueurs votent pour accepter ou refuser le gouvernement.



— En cas d'acceptation, des lois sont piochées et une loi est adoptée.

6.2.3 Conditions de victoire

La partie se termine lorsqu'une des conditions suivantes est atteinte :

— Les Libéraux gagnent si cinq lois libérales sont adoptées ou si Hitler est éliminé.

— Les Fascistes gagnent si six lois fascistes sont adoptées ou si Hitler est élu Chancelier après l'adoption de trois lois fascistes.