



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.git>

Table des matières

1	Introduction	3
1.1	Contexte	3
1.2	Problématique et objectifs	3
2	Environnement de travail	4
2.1	Langages et technologies	4
2.2	Outils de développement	4
2.3	Graphes et visualisation	5
2.4	Utilisation de l'intelligence artificielle	6
2.4.1	Intégration de l'API Gemini (Moteur de Jeu)	7
3	Description du projet et objectifs	8
3.1	Fonctionnement général du jeu	8
3.2	Objectifs pédagogiques et techniques	8
4	Difficultés rencontrées et Solutions Techniques	9
4.1	La Modélisation et la Visualisation du Graphe de Confiance .	9
4.2	L'Intelligence Artificielle des Bots (Prise de Décision)	10
4.3	Le Rythme du Jeu et la Narration (UX)	11
4.4	Gestion de l'Aléatoire et Mise en Scène	11
4.5	La Synchronisation des États et les Boucles Infinies	12
4.6	La Gestion de l'Audio et l'Autoplay	13
5	Bilan	13
6	Annexes	14
6.1	Exemple d'exécution du projet	14
6.2	Manuel utilisateur	14
6.2.1	Rôles des joueurs	14
6.2.2	Déroulement d'un tour	15
6.2.3	Conditions de victoire	16
7	Conditions Spéciales (Clé API Gemini)	17
7.1	Génération Narrative : La Chute de la République	18
7.2	L'Analyste Stratégique et Narratif (IA)	18

1 Introduction

1.1 Contexte

La figure suivante illustre le contexte global du projet : le développement d'une adaptation web du jeu *Secret Hitler* combinant technologies web modernes, théorie des graphes pour les interactions et assistance par intelligence artificielle.



FIGURE 1 – Contexte du projet : Technologies web, Graphes et IA

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 :

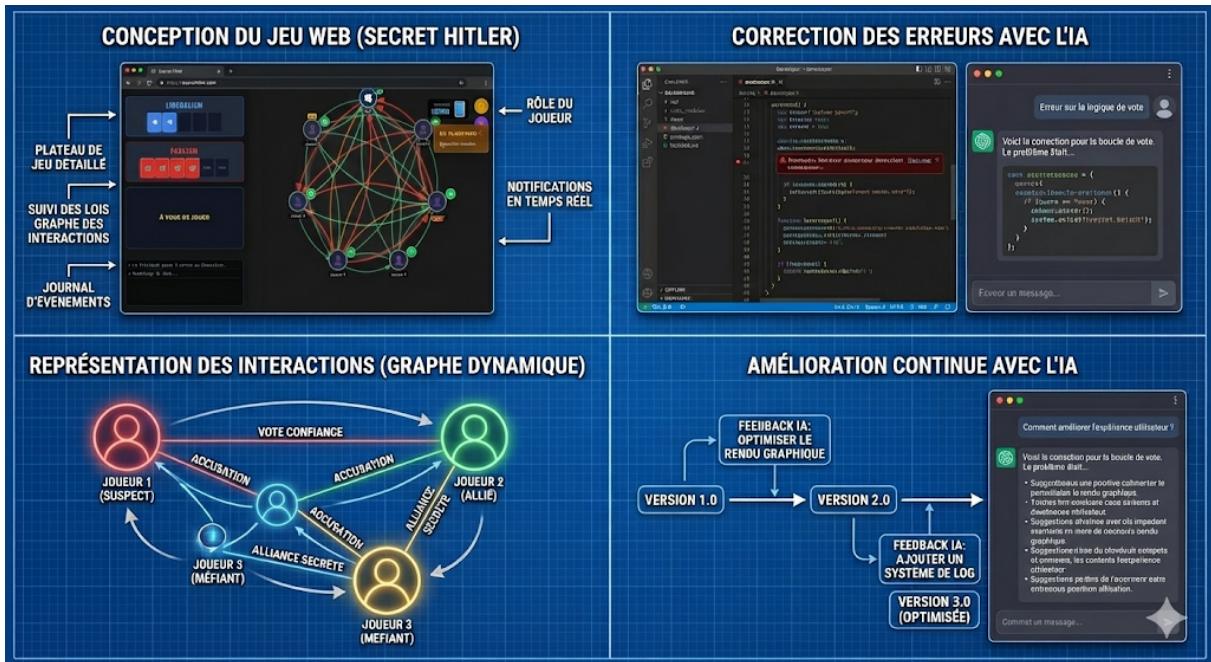
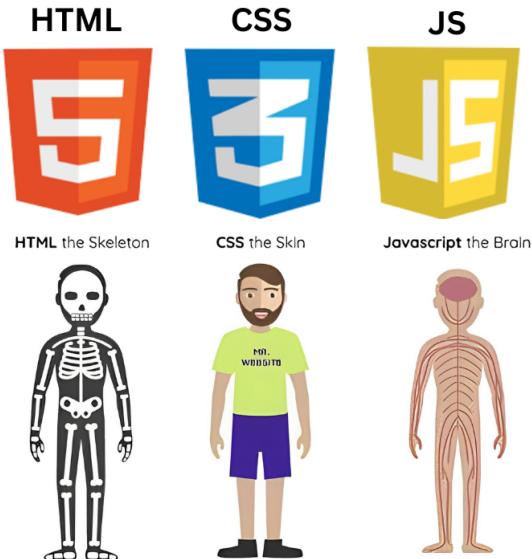


FIGURE 2 – Les quatre objectifs principaux du projet

2 Environnement de travail

2.1 Langages et technologies



2.2 Outils de développement

- Éditeur de code :



— Navigateurs :



— Gestion de versions :



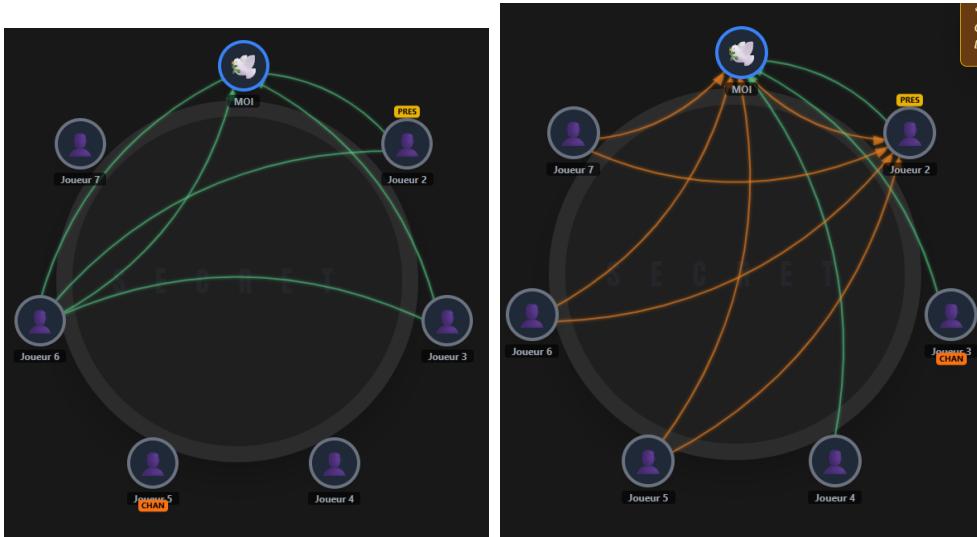
2.3 Graphes et visualisation

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 apparaîtront).



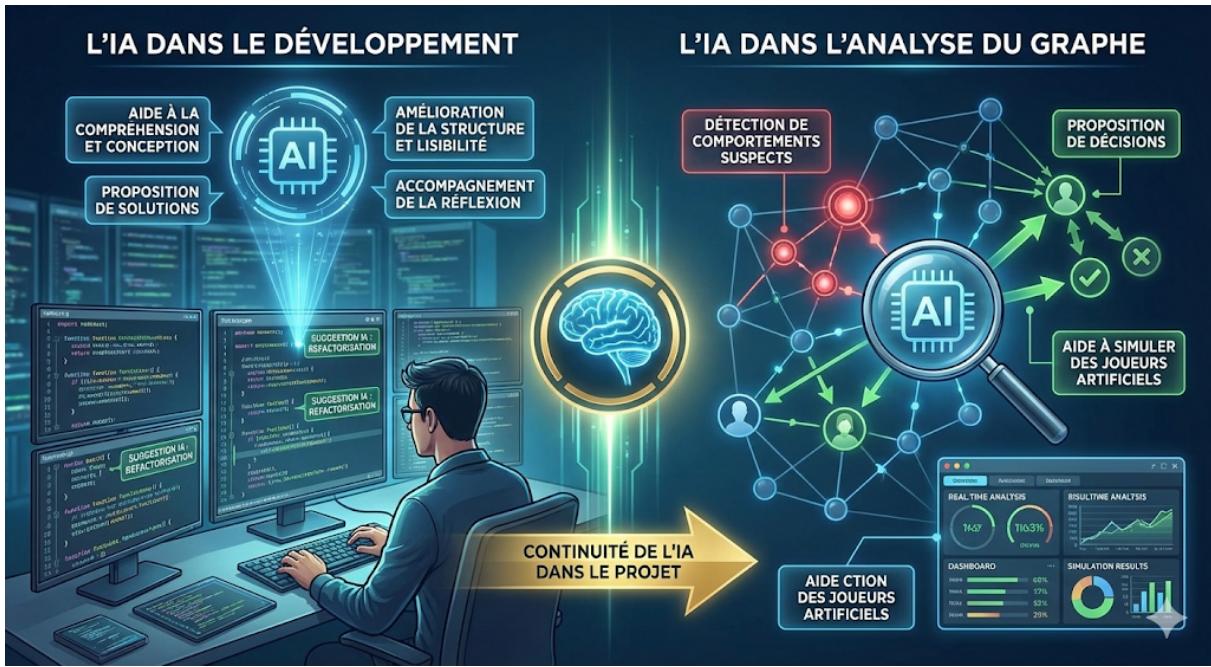
- **Arêtes (liens) :** Une arête entre deux nœuds représente une interaction entre deux joueurs. Exemple : la confiance en vert, la suspicion en orange. Ces arêtes évoluent au cours de la partie selon les actions des joueurs.

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.



La figure ci-dessous synthétise le double rôle de l'intelligence artificielle dans ce projet : elle agit d'abord comme un levier de productivité pour l'équipe de développement, puis intervient au cœur du jeu pour analyser les graphes et simuler des comportements.



2.4.1 Intégration de l'API Gemini (Moteur de Jeu)

Au-delà de l'aide au développement, nous avons intégré l'API **Google Gemini** directement dans le moteur du jeu pour animer les bots. Cette intégration permet aux joueurs artificiels de prendre des décisions complexes et de justifier leurs votes dans le chat.

Fonctionnement technique : Le système repose sur un cycle de requête asynchrone :

- Contextualisation** : Le jeu génère un fichier JSON contenant l'état actuel de la partie (historique des votes, lois promulguées, suspicions du graphe).
- Prompt Engineering** : Ce contexte est envoyé à l'API avec une consigne système stricte définissant la personnalité du bot (ex : "*Tu es un Libéral méfiant qui doute du Chancelier*").
- Décision** : L'API renvoie une action de vote (JA/NEIN) et une phrase de dialogue immersive.

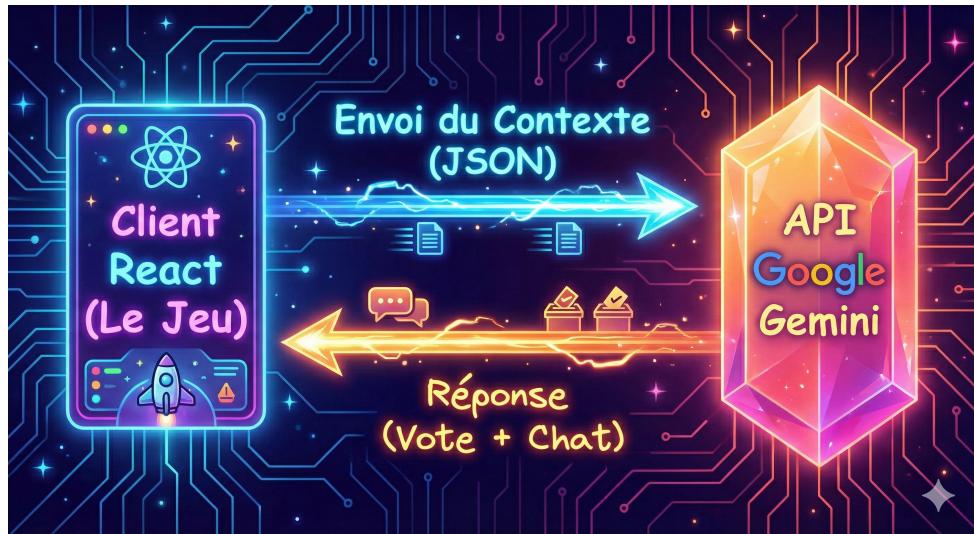


FIGURE 3 – Flux de données entre le jeu Secret Hitler et l’API Gemini

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. La figure ci-dessous synthétise les quatre piliers fondamentaux de ce travail, alliant compétences techniques, modélisation mathématique et méthodologie collaborative.



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 4).

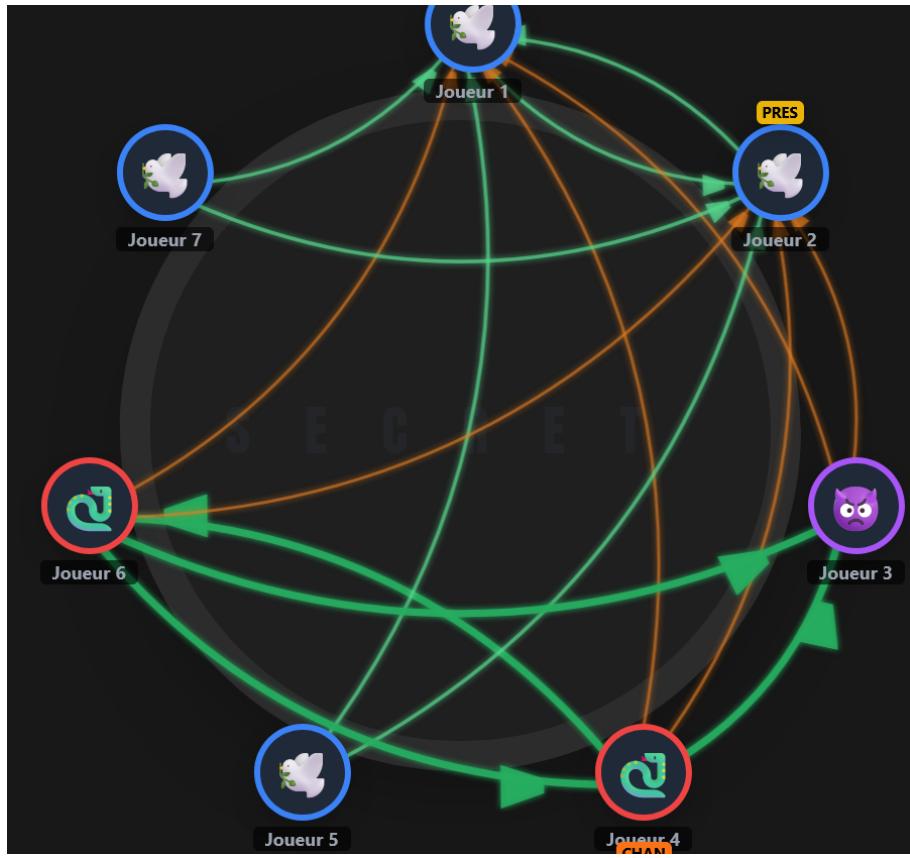


FIGURE 4 – Définition technique des arêtes pondérées

De plus, la logique du jeu influence directement ces couleurs (orange, vert) (Figure 5).

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 suspitions (lignes orange) se forment naturellement en fonction des actions de chaque gouvernement.

'ons ou créons ensemble

FIGURE 5 – 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 6 – 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.

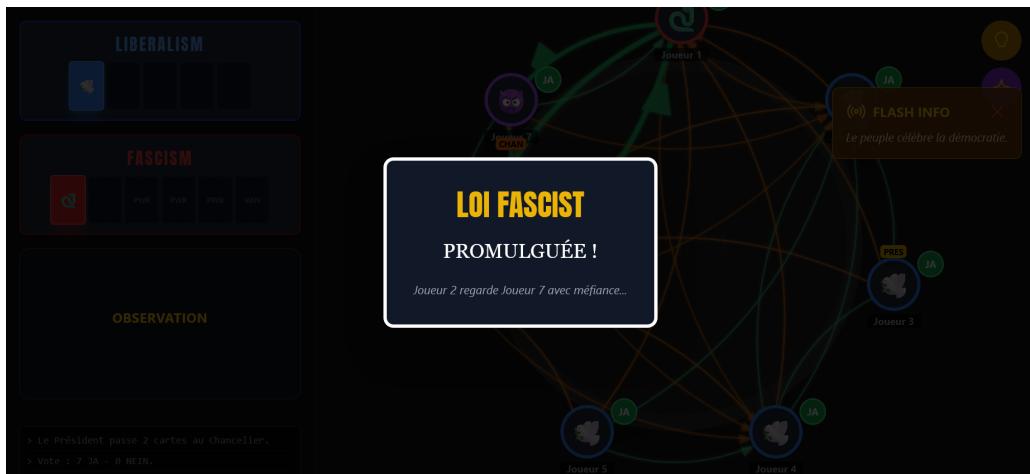


FIGURE 7 – Overlay d'annonce : Une loi Fasciste est promulguée

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 8 – Attribution Libéral

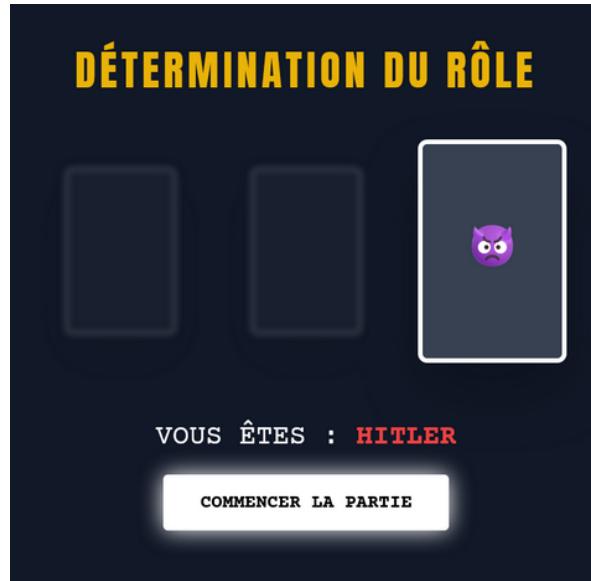


FIGURE 9 – Attribution Hitler

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 10 – 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 coupent la musique.

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



FIGURE 11 – Écran d'accueil forçant l'interaction utilisateur pour activer l'audio

(Activation Audio Requise)

FIGURE 12 – 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

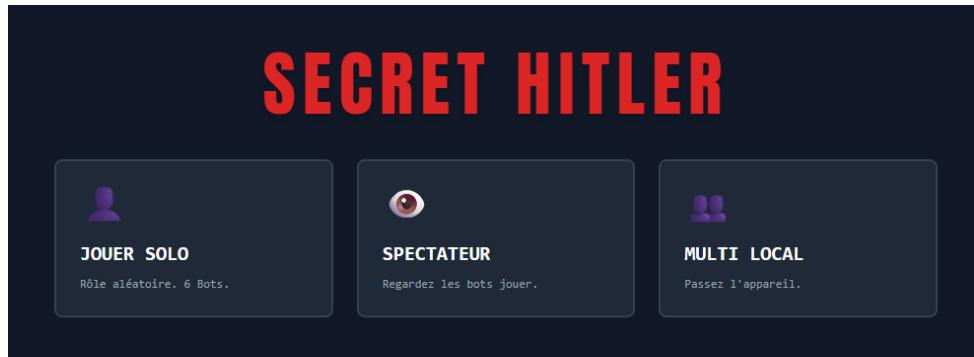


FIGURE 13 – Lancement du jeu

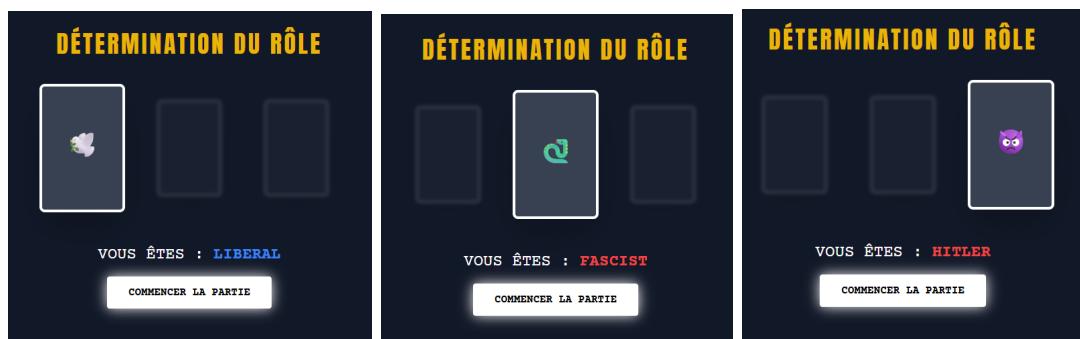


FIGURE 14 – Détermination du rôle



FIGURE 15 – 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 :

```

> Le Président passe 2 cartes au Chancelier.
> Vote : 7 JA - 0 NEIN.
> Joueur 3 propose Joueur 6 comme Chancelier.
> NOUVEAU TOUR
> -----
>  Une loi FASCISTE est promulguée !

```

FIGURE 16 – Le déroulement du jeu



FIGURE 17 – 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 :

- **Victoire des Libéraux** : Ils gagnent si 5 lois libérales sont adoptées ou si Hitler est éliminé.



FIGURE 18 – Tableau des lois Libérales



FIGURE 19 – Écran de Victoire Libérale

- **Victoire des Fascistes** : Ils gagnent si 6 lois fascistes sont adoptées.



FIGURE 20 – Tableau des lois Fascistes



FIGURE 21 – Écran de Victoire Fasciste

7 Conditions Spéciales (Clé API Gemini)

Bien que le cœur du jeu "Secret Hitler" soit entièrement fonctionnel de manière autonome, le projet intègre une couche optionnelle d'Intelligence Artificielle générative via l'API Google Gemini. L'ajout d'une clé API enrichit l'immersion narrative et l'interaction stratégique.

7.1 Génération Narrative : La Chute de la République

Il existe une condition de victoire alternative et immédiate pour les Fascistes : si Hitler est élu Chancelier après que 3 lois fascistes ont déjà été votées, la partie s'arrête instantanément.

Au lieu d'un simple message de fin, l'IA prend le relais pour rédiger un article de journal unique, annonçant la chute tragique de la République et l'instauration du nouvel ordre. Cela rend chaque conclusion historiquement marquante.

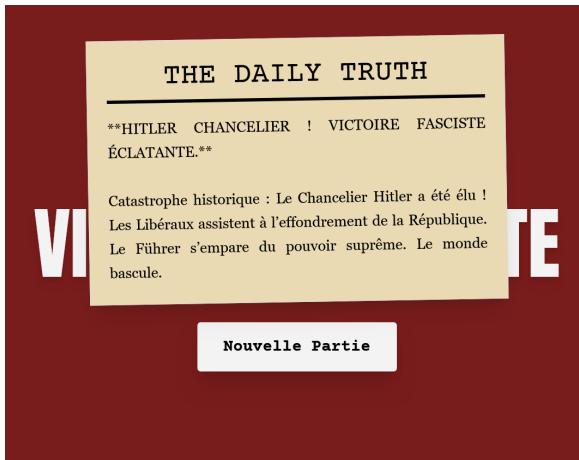


FIGURE 22 – Exemple de variation d'article généré par Gemini.

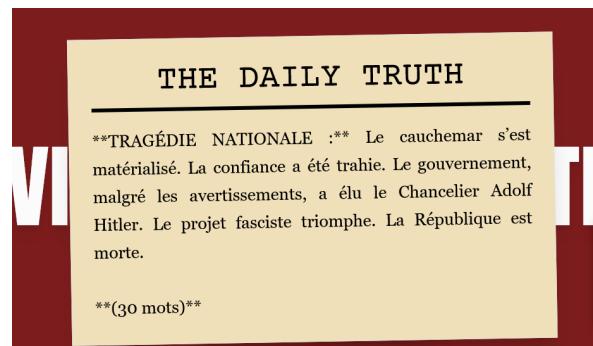


FIGURE 23 – Article : "Tragédie Nationale" généré par l'IA

7.2 L'Analyste Stratégique et Narratif (IA)

Une fonctionnalité d'assistance est accessible en cours de partie via le bouton violet et les événements de jeu. L'IA analyse la situation en temps réel pour offrir deux types d'interventions :

- **Le Conseil Stratégique** : En cliquant sur l'icône, l'IA analyse les logs récents (votes, lois) et suggère une stratégie adaptée au rôle du joueur (ex : se fondre dans la masse pour un fasciste).
- **La Narration Contextuelle** : Lorsqu'une loi passe, l'IA génère un "Flash Info" unique sous forme de propagande radio, renforçant l'immersion.



FIGURE 24 – Conseil tactique généré par l'IA pour un Fasciste.

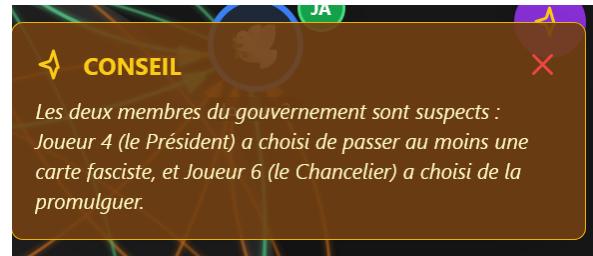


FIGURE 25 – Flash Info narratif réagissant à une loi votée.

P.S. : Je vous jure que je suis Libéral.

(Mais c'est exactement ce que dirait Hitler...)