

« Utilisation des API d'OpenAI autour de ChatGPT pour la création d'un assistant vocal »



Réalisé par : Salah Eddine Aboukacim - Mohamed Ghoul - Abdellah Adansar - Anas Balrhi - Zouhair Dkhissi - Ayoub El Alami El Filali

Encadré Par : M.Michel Winter

Année universitaire : 2023/2024

Remerciement

Nous tenons à exprimer notre profonde gratitude envers tous ceux qui ont contribué à la réalisation de ce projet. Tout d'abord, nous souhaitons remercier Monsieur Michel WINTER, notre tuteur professionnel à l'université, pour son précieux soutien et son implication tout au long du projet. Sa guidance a été essentielle pour mener à bien ce travail.

Nous adressons également nos remerciements chaleureux à Monsieur Leo DONATI, Professeur de l'UCA, pour son évaluation de notre travail et ses précieux conseils. Sa contribution inlassable tout au long de l'année universitaire a été d'une grande valeur pour nous.

Nous souhaitons également exprimer notre reconnaissance envers tout le corps professoral de notre école pour leurs éclairages et leurs conseils. Leurs connaissances et leur expertise ont grandement enrichi notre travail.

Enfin, nous tenons à remercier tous les membres de l'équipe impliqués dans ce projet pour leur collaboration et leur dévouement. Leurs efforts conjoints ont été essentiels pour les réalisations que nous avons accomplies.

Nous sommes reconnaissants envers toutes ces personnes pour leur soutien et leurs contributions, qui ont été déterminantes dans la réussite de ce projet.

SOMMAIRE

1.Introduction	4
1.1 Objectif du Projet	4
1.2 Fonctionnalités Principales :	5
1.2.1 Modélisation :	6
1.2.1.1 Diagramme de cas d'utilisation	6
1.2.1.2 Diagramme de contexte	6
1.2.1.3 Diagramme de séquence	7
1.3 Structure du Projet:	7
1.4 Langages et Technologies Utilisées	8
2.Interface Utilisateur :	9
3.Fonctionnalité du Chatbot	10
3.1 Reconnaissance Vocale :	10
3.2 Communication avec ChatGPT :	14
3.3 Interaction entre l'affichage et ChatGPT :	15
3.4 Intégration des Réponses Vocales :	16
4.Évolution du Projet et Problèmes rencontrés	17
4.1 Version 0:	17
4.2 Version 1:	20
4.3 Version 2:	22
4.4 Version 3:	24
4.5 Version 4:	26
4.6 Version 5	29
5-Gestion des tâches :	32
5-1: Diagramme de Gantt	32
5-2: Diagramme de Pert	32
6. Perspective et Amélioration :	33
7.Conclusion :	34
8-Glossaire :	35
9-WEBOGRAPHIE:	36

1.Introduction

Le projet vise à exploiter les API d'OpenAI autour de ChatGPT pour créer un assistant vocal. ChatGPT, efficace dans le traitement du langage naturel, nécessite l'intégration du mécanisme de "Function Calling" pour être utilisé en tant qu'interface homme-machine. L'objectif est de se familiariser avec ces API, notamment Whisper pour la transcription audio en texte, et de développer un démonstrateur.

Ce démonstrateur sera un logiciel contrôlable par la voix, chargé de gérer le score des joueurs lors de jeux de dés ou de cartes. Les fonctionnalités orales incluent la création de parties, l'ajout de points, la lecture du score des joueurs, et la déclaration de fin de partie.

Les tâches à réaliser se divisent en trois thématiques : la transcription Speech-to-Text avec des tests de prototypes et la comparaison entre Whisper et 'webkitSpeechRecognition', le GPT Function Calling avec des tests de prototypes et la définition de fonctions, et enfin, le tableau des scores avec l'implémentation des fonctions pour GPT et l'affichage en HTML/CSS.

Le projet sera développé en JavaScript, sans l'utilisation a priori de frameworks ou bibliothèques externes.

Ce projet offre une opportunité concise d'explorer les capacités de ChatGPT en tant qu'assistant vocal, avec une focalisation sur la transcription audio en texte et l'appel de fonctions pour une interaction vocale complète.

1.1 Objectif du Projet

L'objectif principal du chatbot Jarvis est de développer un assistant vocal avancé en exploitant les API d'OpenAI, notamment ChatGPT. Le but fondamental est d'optimiser l'interaction homme-machine en permettant à ChatGPT de détecter et d'exécuter des commandes spécifiques à partir du langage naturel de l'utilisateur, grâce au mécanisme de "Function Calling" fourni par OpenAI.

Plus précisément, les objectifs spécifiques incluent :

Intégration d'OpenAI : Assurer une intégration fluide de ChatGPT dans Jarvis en utilisant le "Function Calling", permettant au chatbot de comprendre et répondre à des instructions complexes.

Transcription Audio en Texte : Utiliser l'API Whisper d'OpenAI pour transcrire les commandes vocales des utilisateurs en texte, facilitant ainsi le traitement par le chatbot.

Démonstrateur Vocal : Développer un démonstrateur fonctionnel qui réagit de manière dynamique aux commandes vocales. Les fonctionnalités clés englobent la gestion de parties de jeux, l'ajout de points, la consultation des scores et la fin de partie.

Contrôle Vocal Intégral : Permettre aux utilisateurs de diriger Jarvis exclusivement par la voix, créant une expérience immersive et éliminant la nécessité d'interactions manuelles.

1.2 Fonctionnalités Principales :

Le chatbot offre plusieurs fonctionnalités clés, dont notamment :

Interface Utilisateur Intuitive : L'interface utilisateur comprend un tableau de joueurs et une barre de défilement pour naviguer dans l'historique de la conversation, offrant ainsi une expérience utilisateur simple et intuitive.

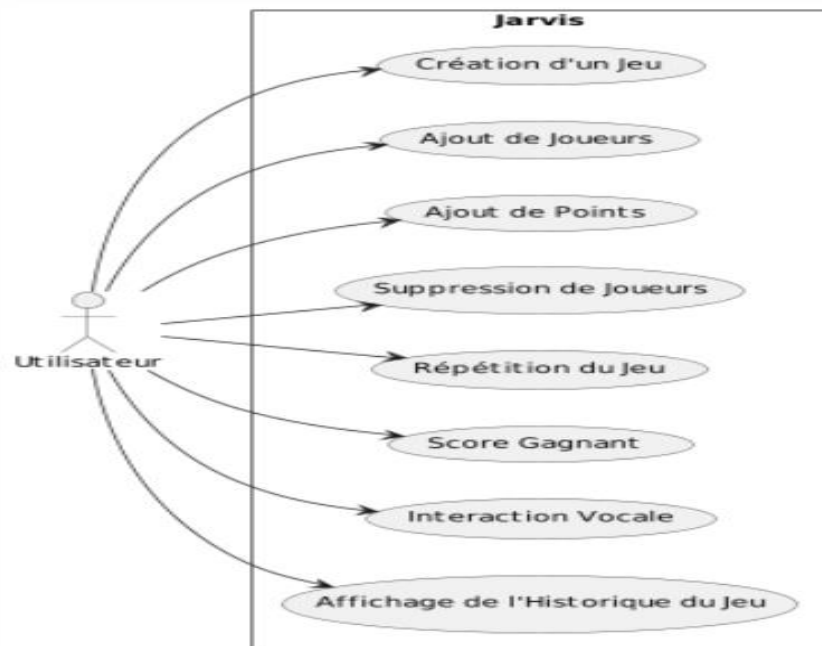
Reconnaissance vocale : Les utilisateurs peuvent interagir avec Jarvis en utilisant la reconnaissance vocale, simplifiant ainsi le processus d'entrée des commandes.

Communication avec ChatGPT : Le modèle d'intelligence artificielle ChatGPT est intégré pour comprendre et répondre aux commandes de l'utilisateur, offrant une interaction conversationnelle avancée.

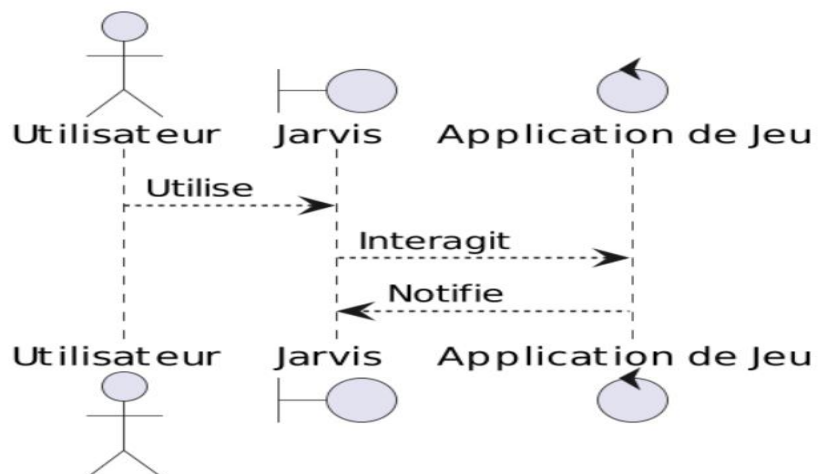
Gestion de l'Histoire du Jeu : Jarvis maintient un historique de la conversation et de l'évolution du jeu, facilitant la compréhension du contexte des commandes de l'utilisateur.

1.2.1 Modélisation :

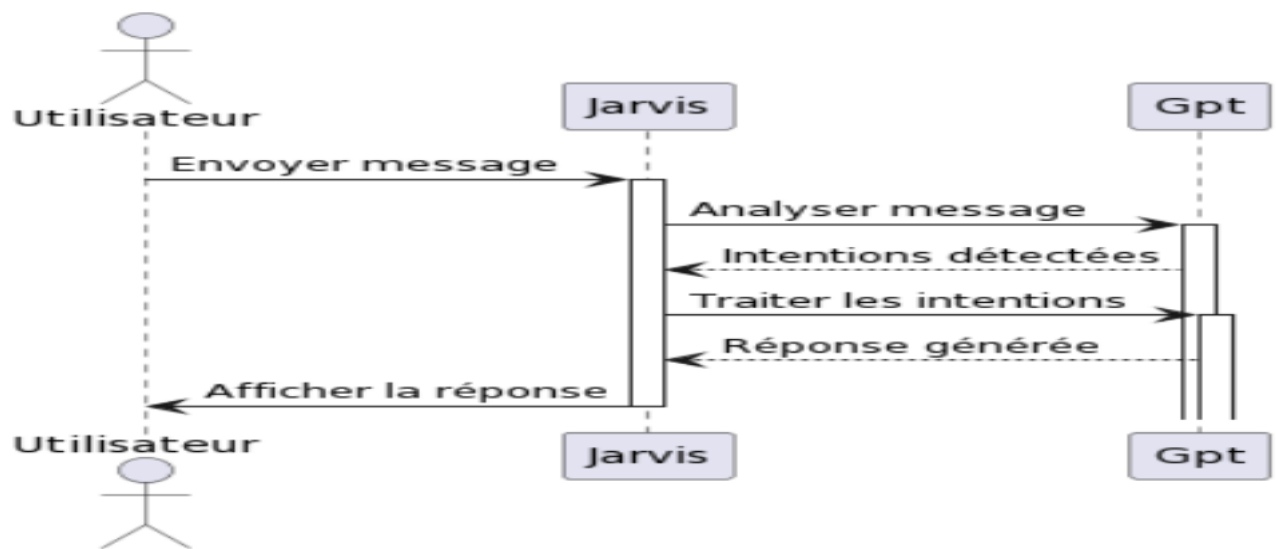
1.2.1.1 Diagramme de cas d'utilisation



1.2.1.2 Diagramme de contexte



1.2.1.3 Diagramme de séquence



1.3 Structure du Projet:

Le code du projet est structuré en HTML, CSS et JavaScript. L'interface utilisateur est construite en HTML et CSS, assurant une présentation claire et esthétique. La partie fonctionnelle du chatbot est mise en œuvre en JavaScript, avec une utilisation significative de la reconnaissance vocale et de la synthèse vocale pour améliorer l'expérience utilisateur.

La structure du projet Jarvis Chatbot repose sur une approche modulaire, avec une répartition claire des fichiers pour garantir une gestion efficace des fonctionnalités. Les technologies utilisées incluent HTML, CSS, et JavaScript pour construire une interface utilisateur intuitive et fonctionnelle.

index.html : Le fichier principal HTML qui constitue l'entrée de l'application. Il incorpore les autres fichiers HTML et définit la mise en page générale de l'interface.

styles.css : Fichier de style principal qui assure la présentation visuelle cohérente de l'interface utilisateur.

index.js : Fichier JavaScript principal qui regroupe la logique métier du chatbot, gère la reconnaissance vocale, la communication avec ChatGPT, et la synthèse vocale.

Organisation de la Logique Métier: La logique métier du chatbot est modulaire et organisée en fonctions distinctes. Par exemple, la gestion des points des joueurs, la mise à jour de l'historique, et la communication avec ChatGPT sont isolées dans des fonctions spécifiques pour une maintenance simplifiée.

Les interactions vocales et la reconnaissance des commandes sont gérées de manière à optimiser la compréhension et la réactivité du chatbot.

1.4 Langages et Technologies Utilisées

Le projet fait usage des technologies suivantes :

1-Chatgpt OpenAi :

ChatGPT de OpenAI est un modèle de langage avancé qui permet aux développeurs d'intégrer des capacités de chatbot dans leurs applications. Il peut comprendre et générer du texte cohérent pour interagir avec les utilisateurs. L'API fournie par OpenAI permet d'accéder à ce modèle et de l'utiliser pour créer des chatbots personnalisés.



2–Html et Css :

HTML (HyperText Markup Language) et CSS (Cascading Style Sheets) sont des langages fondamentaux utilisés pour développer des sites web. HTML est utilisé pour structurer le contenu d'une page, tandis que CSS est utilisé pour styliser et mettre en forme ce contenu. HTML définit la structure logique des éléments de la page, tandis que CSS permet de personnaliser l'apparence visuelle des éléments. En utilisant HTML et CSS ensemble, les développeurs peuvent créer des sites web interactifs et attrayants sur le plan esthétique.



3- Javascript :

La logique métier, telle que la gestion du flux de conversation, des jeux et de l'historique, peut être implémentée en utilisant JavaScript. JavaScript permet de définir les règles et les comportements spécifiques du chatbot, d'interagir avec les données utilisateur, de manipuler les réponses et de prendre des décisions en fonction des entrées de l'utilisateur. En utilisant JavaScript, vous pouvez personnaliser et contrôler le comportement du chatbot pour répondre aux besoins spécifiques de votre application.

2.Interface Utilisateur :

L'interface utilisateur est conçue en utilisant du HTML et du CSS. L'interface présente un design épuré et une disposition claire, facilitant la compréhension et l'utilisation pour les utilisateurs, les choix de couleurs contribuent à une expérience visuelle agréable.

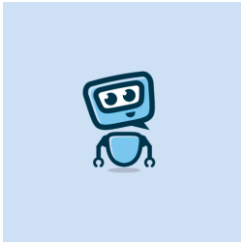
La structure HTML est organisée en deux parties, chacune délimitant des zones spécifiques de l'interface.

La première partie contenant des éléments clés tels que les boutons, les différentes sections du contenu et un tableau d'affichage des scores des joueurs permettant à l'utilisateur de percevoir les informations, tandis que la seconde partie affiche un l'historique de conversation entre l'utilisateur et l'assistant vocal Jarvis.

L'utilisateur a la possibilité d'activer et désactiver la vocalisation de Jarvis à l'aide d'une case à cocher ainsi qu'une autre case à cocher permettant de définir le déclenchement de la reconnaissance vocale par la touche espace ou par un mot déclencheur. Un bouton "Recommencer" permettant de réinitialiser la partie ainsi que le tableau des scores qui se met à jour suite à l'avancement de la partie.

L'historique de l'interaction utilisateur-Jarvis avec une barre de défilement pour parcourir l'historique des messages. Nous avons également une rubrique aides contenant une page FAQ pour répondre aux questions liées à l'utilisation de l'application et une seconde page pour les signaux de bugs.

3.Fonctionnalité du Chatbot



La fonctionnalité de reconnaissance vocale est cruciale pour le Chatbot, offrant une interaction naturelle et pratique. Elle élargit l'accessibilité en permettant aux utilisateurs de communiquer vocalement, améliorant ainsi l'expérience utilisateur. Cette fonctionnalité favorise une interaction plus fluide et rapide, particulièrement utile dans des situations où la saisie de texte est peu pratique. L'intégration de la reconnaissance vocale avec l'API OpenAI renforce l'efficacité de la communication, permettant au Chatbot de traduire les entrées vocales en texte pour générer des réponses pertinentes. En ajoutant des réponses vocales, l'expérience utilisateur devient encore plus immersive, créant une interaction conversationnelle et personnalisée. En résumé, la reconnaissance vocale améliore significativement l'expérience utilisateur du Chatbot, ouvrant des perspectives d'utilisation diverses et créant une interaction plus engageante.

3.1 Reconnaissance Vocale :

La reconnaissance vocale constitue une pierre angulaire de notre projet, offrant à l'assistant vocal la capacité d'interpréter précisément les commandes vocales. Cette technologie avancée améliore l'interaction utilisateur en permettant une communication fluide via la voix. Elle élargit également l'accessibilité en offrant une alternative aux interfaces textuelles, rendant notre assistant plus inclusif.



En intégrant la reconnaissance vocale, notre projet s'appuie sur l'API OpenAI pour enrichir cette fonctionnalité. Cette collaboration permet d'exploiter des modèles linguistiques avancés, comme Whisper, améliorant ainsi la compréhension du contexte des commandes vocales et générant des réponses plus précises et pertinentes. Cette synergie renforce la capacité de notre assistant vocal à fournir des interactions intelligentes et personnalisées.

En outre, l'intégration d'outils complémentaires, tels que Whisper, ajoute une dimension supplémentaire à notre projet. Ces outils offrent des fonctionnalités avancées, comme la suppression du bruit dans les commandes vocales, améliorant ainsi la qualité de l'interaction. En combinant la reconnaissance vocale avec des outils comme Whisper, notre projet vise à offrir une expérience utilisateur optimale, alliant précision, rapidité et intelligence conversationnelle.

Description des APIs de Reconnaissance Vocale :

API JavaScript (API Web Speech) :

Description :



L'API JavaScript occupe un rôle central dans la mise en œuvre de la reconnaissance vocale au sein du chatbot. Cette interface de programmation permet de capturer de manière efficace les commandes vocales émises par l'utilisateur. En intégrant cette API, le chatbot devient capable de traiter la voix comme un moyen d'interaction, ouvrant ainsi la porte à une communication plus naturelle et immersive. Grâce à cette technologie, les utilisateurs peuvent exprimer leurs besoins, poser des questions, ou donner des commandes simplement en utilisant leur voix, contribuant ainsi à une expérience utilisateur plus fluide et conviviale.

Points forts :

L'API JavaScript pour la reconnaissance vocale présente plusieurs atouts essentiels dans le contexte du chatbot. En particulier, elle excelle dans la détection immédiate du mot-clé "Jarvis", permettant une activation rapide de la reconnaissance vocale dès que le mot-clé est prononcé. Cette réactivité renforce l'efficacité de l'interaction en offrant une réponse instantanée à l'utilisateur. De plus, la possibilité de spécifier la langue constitue un avantage crucial, améliorant la précision de la reconnaissance vocale. En évitant les confusions liées aux variations d'accent, cette fonctionnalité assure une compréhension plus précise des commandes vocales, contribuant ainsi à une expérience utilisateur optimisée et sans ambiguïtés. En combinant ces points forts, l'API JavaScript crée une base solide pour une reconnaissance vocale performante au sein du chatbot.

API Whisper (OpenAI) :

Description :

L'API Whisper, développée par OpenAI, est intégrée pour assurer la transcription audio au sein de notre projet. Cette interface de programmation se distingue par sa capacité à convertir de manière précise les données audio en texte, offrant ainsi une fonctionnalité cruciale pour la compréhension des interactions vocales. En utilisant l'API



Whisper, notre projet est en mesure de traiter des fichiers audio, améliorant ainsi la polyvalence de notre système en prenant en charge la transcription de conversations, de messages vocaux, et d'autres formes d'entrées audio.

Points forts :

L'API Whisper présente une précision exceptionnelle, particulièrement lorsque la prononciation est claire. Cela en fait un outil idéal pour la transcription audio dans des contextes où la clarté et la précision sont essentielles. En mettant l'accent sur la compréhension des nuances vocales, Whisper excelle dans la conversion précise de l'audio en texte, garantissant une interprétation fidèle des commandes vocales. Cette précision accrue contribue à une meilleure qualité de la transcription, permettant à notre projet de fournir des réponses plus pertinentes et adaptées aux intentions de l'utilisateur. Ainsi, l'intégration de l'API Whisper enrichit significativement la capacité de notre système à traiter et comprendre les données audio, renforçant ainsi la qualité globale de l'expérience utilisateur, on peut citer aussi le traitement de plusieurs langues donc Les utilisateurs peuvent interagir dans leur langue préférée



Comparaison des API de Reconnaissance Vocale :

Détection du Mot-Clé "Jarvis" :

API JavaScript : Détecte immédiatement le mot-clé "Jarvis" dès que le mot-clé est prononcé, permettant à l'utilisateur d'envoyer directement son message et par conséquent un délai de réponse court .

API Whisper : Nécessite 2 à 3 secondes d'attente après l'événement vocal, ce qui peut entraîner un léger délai avant l'envoi du message.

Précision et Gestion des Accents :

API JavaScript : Permet de spécifier la langue, améliorant la précision en prenant en compte les variations d'accent des utilisateurs.

API Whisper : Excellente précision dans la transcription audio, même avec des accents, mais n'offre pas la possibilité de spécifier la langue, ce qui entraîne des réponses hybrides avec deux langues différentes si il reconnaît un accent spécifique.

Coût et Sécurité :

API JavaScript : Considéré comme meilleur en termes de coût et de sécurité.

API Whisper : Offre une précision supérieure mais peut être plus coûteux et nécessite une attention particulière à la sécurité.

Choix des APIs :

Lors du développement de notre site web, nous avons pris la décision stratégique d'utiliser l'API Web Speech pour la détection du mot-clé "Jarvis". Cette API s'est révélée plus efficace dans la reconnaissance précise du mot-clé, assurant une réponse rapide et fiable aux commandes vocales de l'utilisateur. Cependant, étant donné que l'API Web Speech précise la langue d'interaction, nous avons intégré l'API Whisper pour la reconnaissance vocale associée à la barre d'espace. Cela nous a permis d'offrir une expérience utilisateur homogène en utilisant une combinaison optimale de ces deux technologies, offrant ainsi une interaction fluide et globale sur notre site web

Implémentation des APIs de Reconnaissance Vocale dans le Code :

Reconnaissance Vocale (Web Speech API) :

Le code utilise l'API Web Speech pour la reconnaissance vocale.

La fonction `DitLaVoix()` est appelée lorsqu'il y a des résultats de la reconnaissance vocale.

Lorsque le mot-clé "Jarvis" est détecté avec une confiance suffisante, le code extrait la commande vocale et la traite.

Enregistrement Audio et API Whisper :

L'enregistrement audio commence lorsque la barre d'espace est pressée et s'arrête lorsque la barre d'espace est relâchée.

L'audio enregistré est transmis à l'API Whisper pour la transcription.

La fonction `invokeWhisperAPI()` envoie l'audio à l'API Whisper et traite les résultats.

3.2 Communication avec ChatGPT :

La communication avec ChatGPT dans le projet se fait de manière asynchrone, permettant une interaction fluide entre l'utilisateur et l'assistant virtuel. Voici les principales étapes de la communication avec ChatGPT :

Requêtes vers GPT :

Lorsqu'un utilisateur interagit avec l'interface utilisateur, des requêtes sont envoyées à l'API GPT pour obtenir des instructions ou des réponses en fonction des actions de l'utilisateur.

Les requêtes incluent des informations telles que les commandes vocales, les actions de l'utilisateur, ou d'autres données pertinentes nécessaires pour la génération de réponses cohérentes.

Réponses de GPT :

Une fois la requête envoyée, l'application attend une réponse de GPT. Les réponses peuvent inclure des instructions pour le jeu, des informations sur les joueurs, des conseils, ou d'autres éléments pertinents.

Les réponses sont traitées et interprétées pour mettre à jour l'interface utilisateur et l'historique du jeu.

Gestion des Erreurs :

En cas d'erreurs ou de problèmes lors de la communication avec GPT, des mécanismes de gestion d'erreur sont mis en place pour assurer la stabilité de l'application.

Les erreurs éventuelles, telles que des requêtes non réussies, sont consignées et gérées pour éviter des interruptions majeures dans l'expérience utilisateur.

Rôle Spécifique de GPT :

GPT joue un rôle central dans la génération de contenu textuel, fournissant des instructions, des réponses verbales, et contribuant ainsi au déroulement du jeu.

Les réponses de GPT sont souvent dynamiques, s'adaptant aux actions spécifiques de l'utilisateur et aux étapes du jeu.

3.3 Interaction entre l'affichage et ChatGPT :

L'interaction entre l'affichage et ChatGPT est soigneusement orchestrée pour garantir une expérience utilisateur cohérente et immersive. Voici comment cette interaction est mise en œuvre :

Mise à Jour Dynamique de l'Interface :

Lorsqu'une réponse est reçue de GPT, l'interface utilisateur est dynamiquement mise à jour pour refléter les informations fournies.

Les éléments de l'interface, tels que le tableau des joueurs, l'historique du jeu, et d'autres composants, sont ajustés en fonction du contexte du jeu.

Coordination avec les Actions de l'Utilisateur :

L'interface utilisateur prend en compte les actions de l'utilisateur, que ce soit l'ajout de points, la création d'un nouveau jeu, ou d'autres commandes spécifiques.

Les actions de l'utilisateur sont interprétées et transmises à GPT pour obtenir des réponses contextualisées.

Rétroaction Visuelle et Verbale :

En plus des réponses verbales, l'interface utilisateur fournit également une rétroaction visuelle pour renforcer l'interaction. Par exemple, des messages d'instructions, des indicateurs visuels, ou des animations peuvent être utilisés.

La synthèse vocale est coordonnée avec les mises à jour visuelles pour créer une expérience multimodale et engageante.

Synchronisation Temporelle :

L'interaction entre l'affichage et ChatGPT est synchronisée temporellement pour éviter tout décalage perceptible. Les réponses verbales sont déclenchées au bon moment en fonction du contexte du jeu et des actions de l'utilisateur.

3.4 Intégration des Réponses Vocales :

L'intégration des réponses vocales au sein du projet a pour objectif principal d'améliorer l'expérience utilisateur en permettant au système de fournir des réponses verbales aux utilisateurs. La synthèse vocale est présentée comme la méthode clé pour délivrer ces réponses, et elle est implémentée à travers l'utilisation de l'API SpeechSynthesis.

L'API SpeechSynthesis est une interface du langage JavaScript qui permet d'intégrer des fonctionnalités de synthèse vocale. Cette API facilite la création et la manipulation de contenu vocal directement dans les navigateurs web, offrant ainsi une expérience utilisateur plus riche et interactive. L'API utilise principalement l'objet SpeechSynthesis Utterance pour représenter le contenu vocal à synthétiser. Cet objet contient des propriétés telles que :

- text: Le texte à synthétiser.
- lang: La langue dans laquelle le texte doit être prononcé.
- volume: Le volume de la synthèse vocale.
- rate: La vitesse de la synthèse vocale (1 correspond à la vitesse normale).
- pitch: Le ton de la synthèse vocale.

L'API SpeechSynthesis est aussi largement prise en charge par les navigateurs modernes tels que Chrome, Firefox, Safari, et Edge. Cependant, la disponibilité des voix et certaines fonctionnalités peuvent varier d'un navigateur à l'autre.

Dans notre projet la fonction “speakResponse” prend en paramètre le texte à synthétiser ainsi que la langue par défaut, qui est définie comme le français, la synthèse vocale est gérée à travers l'objet “SpeechSynthesisUtterance” comme cité ci-dessus puis va vérifier les différents paramètres pour la personnalisation de la réponse vocal. La fonction assure également l'annulation de toute synthèse vocale précédente avant de déclencher la lecture de la nouvelle réponse.

4.Évolution du Projet et Problèmes rencontrés

4.1 Version 0:

Objectifs de la Version 0:

- **Interface Utilisateur :**

Concevoir et mettre en place une interface utilisateur conviviale.

Intégrer des éléments interactifs pour créer un jeu, ajouter des joueurs et afficher les résultats.

- **Communication avec GPT:**

- Établir une communication avec le modèle GPT.

- Mettre en place les fonctions nécessaires pour envoyer des requêtes au modèle et recevoir les réponses.

- Implémenter une fonction qui initialise un nouveau jeu. Cette fonction peut définir les paramètres initiaux du jeu.

- Mettre en place une fonction qui permet d'ajouter des joueurs au jeu en cours.

- Utiliser un tableau ou une structure de données adaptée pour stocker les informations des joueurs, telles que leurs noms et points.

- Effectuer des tests réguliers pour vous assurer que chaque fonctionnalité fonctionne correctement.

- Débuguer les erreurs qui pourraient survenir pendant le développement.

Fonctionnalités Implémentées:

- **Interface Utilisateur :**

Une interface utilisateur simple a été créée avec un champ d'entrée permettant la communication avec le chatbot. Les utilisateurs peuvent saisir des commandes pour effectuer des actions telles que la création d'un jeu, par exemple : "Créer un jeu avec Winter et Donati", ou pour ajouter des points à un joueur, par exemple : "ajouter 8 points à Winter". Ces commandes sont gérées par deux fonctions distinctes.

- **Fonctions Calling :**

Les fonctions 'create_game' et 'ajout_players' ont été mises en œuvre pour initier un nouveau jeu et ajouter des joueurs, respectivement. Ces fonctions communiquent avec le modèle GPT pour obtenir des instructions sur le déroulement du jeu.

- **Gestion des Données des Joueurs :**

Des problèmes ont été rencontrés dans cette partie, notamment l'affichage. Ce problème va être résolu dans les versions d'après.

- **Communication avec GPT :**

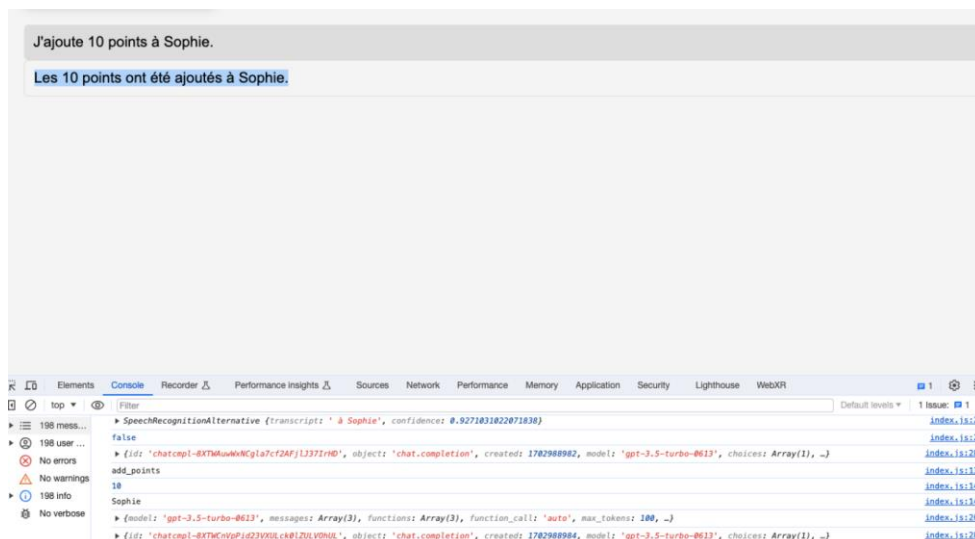
Une fonction 'sendRequest' a été créée pour envoyer des requêtes au modèle GPT. Les réponses de GPT sont interprétées et utilisées pour mettre à jour l'interface.

Challenges Rencontrés:

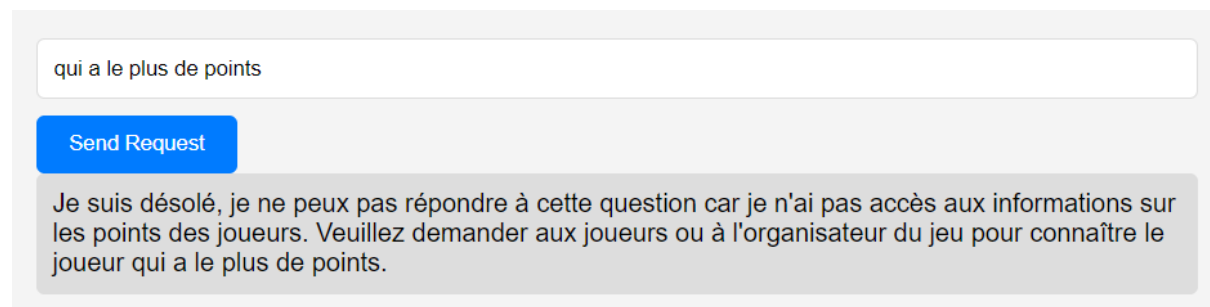
Une des difficultés majeures a été la récupération précise des arguments renvoyés par GPT lors des appels aux fonctions, ce qui a engendré des erreurs dans l'affichage des instructions générées par le modèle.

Par ailleurs, un autre obstacle notable concerne l'affichage du tableau destiné à présenter les joueurs et leurs points. le tableau ne s'affiche pas correctement dans l'interface utilisateur, compromettant la lisibilité et l'expérience globale de l'utilisateur.

Il est également important de noter que le problème d'affichage est attribuable au fait que GPT ne fait pas l'appel à la fonction 'ajouter_joueur', laissant ainsi des arguments non récupérés et non insérés dans le tableau des joueurs.



De plus, un problème notable a été identifié concernant la capacité de l'API GPT à se rappeler des requêtes précédentes. Cette limitation empêche la gestion cohérente et contextuelle des requêtes basées sur des interactions antérieures, limitant ainsi la complexité et la continuité des conversations.



The screenshot shows a chat interface with a light gray background. At the top, there is a white input box containing the text "qui a le plus de points". Below the input box is a blue button with the text "Send Request". Below the button is a gray response box containing the text: "Je suis désolé, je ne peux pas répondre à cette question car je n'ai pas accès aux informations sur les points des joueurs. Veuillez demander aux joueurs ou à l'organisateur du jeu pour connaître le joueur qui a le plus de points."

Un défi supplémentaire réside dans la stabilité des appels à l'API GPT. Des problèmes ont été observés, entraînant des interruptions dans la communication entre l'application et le modèle.

Solutions Apportées:

Introduction d'un Système de Rôle: Un système de rôle a été ajouté pour clarifier les instructions à suivre par GPT, facilitant ainsi l'extraction et l'utilisation des arguments retournés.

Correction de l'Affichage du Tableau: Des corrections ont été apportées pour assurer un affichage correct du tableau des joueurs et de leurs points dans l'interface. Ces modifications vont être implémentées dans la version 1.

Optimisation des appels à GPT: Des ajustements ont été effectués pour améliorer la stabilité des appels à l'API GPT, notamment la gestion des erreurs pour assurer une communication plus fiable.

Tableau historique qui va être ajouté dans la version 3 pour garder l'historique de la conversation avec ChatGPT.

4.2 Version 1:

Objectifs de la Version 1:

L'objectif principal de la version 1 était d'intégrer la reconnaissance vocale dans l'application, offrant ainsi aux utilisateurs la possibilité de communiquer avec l'assistant à l'aide de leur voix. Pour atteindre cet objectif, la version 1 visait à implémenter la fonctionnalité de reconnaissance vocale en utilisant l'API Web Speech. De plus, la mise en place d'un bouton dédié pour démarrer la reconnaissance vocale et l'introduction d'un mécanisme alternatif permettant de déclencher la reconnaissance vocale via la touche espace étaient des composants essentiels de cet objectif.

Fonctionnalités Implémentées:

- ***l'API Web Speech***

Les fonctionnalités mises en œuvre dans cette version comprennent la reconnaissance vocale à l'aide de l'API Web Speech. L'utilisateur peut désormais interagir avec l'assistant en parlant. Pour ce faire, un bouton de démarrage a été créé, permettant d'activer la reconnaissance vocale lorsque celui-ci est activé. Un mécanisme alternatif a également été implémenté, permettant de déclencher la reconnaissance vocale en appuyant sur la touche espace, offrant ainsi une option pratique pour les utilisateurs.

Le code HTML définit une interface utilisateur simple avec un bouton de microphone et une zone pour afficher les résultats de la reconnaissance vocale. Le bouton de démarrage est associé à une fonction JavaScript, 'runSpeechRecog', qui initialise la reconnaissance vocale et écoute les commandes vocales. Lorsque la touche espace est pressée, la fonction est également déclenchée, offrant une alternative ergonomique pour démarrer la reconnaissance vocale.

Notons que le code JavaScript utilise l'API Web Speech et un objet de reconnaissance vocale (webkitSpeechRecognition). Lorsqu'une reconnaissance vocale est détectée, le résultat est affiché dans la zone dédiée, et la requête est envoyée à l'API d'OpenAI pour interprétation. Le code gère également les événements liés à la touche espace pour démarrer et arrêter la reconnaissance vocale de manière fluide.

Challenges Rencontrés:

Implémentation du Bouton Espace: L'implémentation du déclenchement de la reconnaissance vocale via la touche espace a provoqué des erreurs dans la console de débogage.

```
Live reload enabled. index.html:71
✖ Uncaught TypeError: Failed to resolve module specifier "index.js". index.html:1
  Relative references must start with either "/", "./", or "../".
Ecoute en cours ... index.js:19
41 ▶ Uncaught (in promise) DOMException: Failed to execute 'start' on index.js:27
    'SpeechRecognition': recognition has already started.
    at runSpeechRecog (http://127.0.0.1:5500/version1/index.js:27:19)
    at HTMLDocument.<anonymous> (http://127.0.0.1:5500/version1/index.js:32:9)
```

Mise en page du Tableau: Des problèmes persistants liés à la mise en page des informations dans le tableau des joueurs ont été observés après l'intégration de la reconnaissance vocale.

Solutions Apportées:

Récupération des Arguments de GPT: Des ajustements ont été apportés pour résoudre les erreurs dans la console liées à la récupération des arguments de GPT, garantissant une communication fluide.

Correction de la Mise en Page: Des efforts ont été déployés pour résoudre les problèmes de mise en page du tableau des joueurs, améliorant ainsi l'affichage des informations, mais encore un peu de décalage dans le tableau.

<div><div></div><div></div></div>	
Winter	Donati
0	0
10	
	10
10	10

4.3 Version 2:

Objectifs de la Version 2:

La version 2 vise à améliorer significativement la reconnaissance vocale en intégrant l'API Whisper, une technologie destinée à optimiser la qualité et la précision de la compréhension des commandes vocales de l'utilisateur. En parallèle, un ajout essentiel consiste à doter l'assistant de la capacité de répondre vocalement aux commandes de l'utilisateur en utilisant l'interface 'SpeechSynthesisUtterance'. Cette fonctionnalité permettra d'enrichir l'expérience utilisateur en fournissant des réponses audio claires et naturelles, augmentant ainsi l'interactivité globale de l'application.

Fonctionnalités Implémentées:

Reconnaissance Vocale Améliorée avec Whisper API : La version 2 introduit une amélioration significative de la reconnaissance vocale en exploitant l'API Whisper. La fonction `startRecording` est mise en œuvre pour démarrer l'enregistrement audio lorsque l'utilisateur presse la touche espace. Le système enregistre les données audio, les traite via l'API Whisper, et renvoie la transcription textuelle résultante. Cette fonctionnalité renforce la précision de la compréhension des commandes vocales.

Réponse Vocale aux Commandes Utilisateur : La nouvelle fonction `speakResponse` est implémentée pour permettre à l'assistant de répondre vocalement aux commandes de l'utilisateur. Lorsqu'une réponse est générée, l'assistant utilise l'interface `SpeechSynthesisUtterance` pour synthétiser la réponse audio et la restituer à l'utilisateur. Cette fonctionnalité accroît l'interactivité et améliore l'expérience utilisateur en fournissant des réponses claires et naturelles.

Intégration Cohérente : La gestion des événements liés à la touche espace, avec les fonctions `startRecordingOnSpace` et `stopRecordingOnSpace`, assure une intégration fluide de la reconnaissance vocale, permettant à l'utilisateur de démarrer et d'arrêter l'enregistrement de manière intuitive.

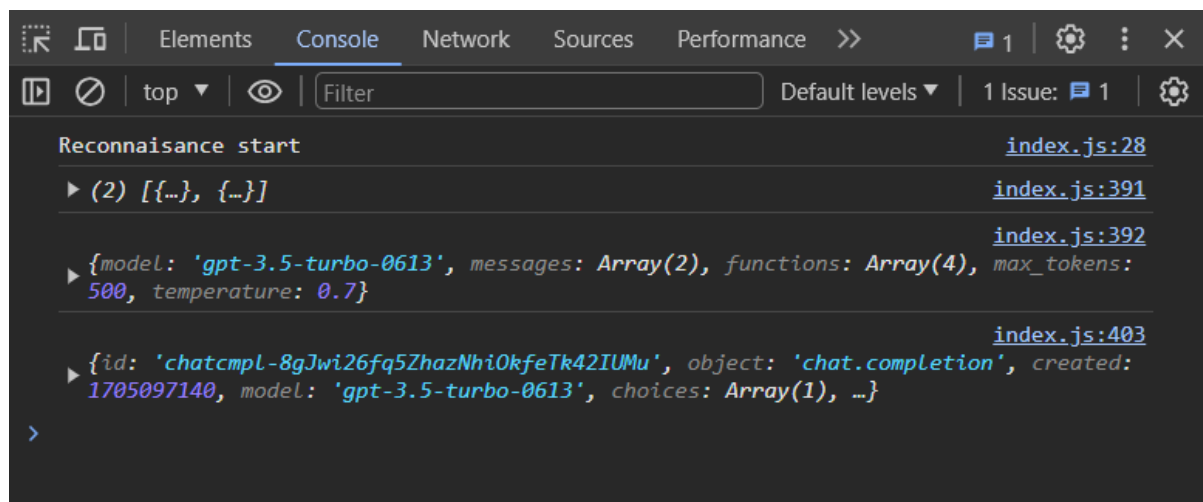
Communication avec l'API et envoi des Requêtes : La fonction `invokeWhisperAPI` est mise en place pour envoyer les données audio à l'API Whisper, récupérer la transcription textuelle, et ensuite déclencher l'envoi de la requête générée à l'API d'OpenAI pour une interaction continue.

Challenges Rencontrés:

La compatibilité des navigateurs est un défi crucial lors de l'implémentation de fonctionnalités audio avancées comme la reconnaissance vocale et la synthèse vocale. Différents navigateurs peuvent avoir des implémentations et des fonctionnalités différentes, ce qui peut affecter la manière dont votre application fonctionne sur différentes plates-formes.

Solutions Apportées:

Une attention particulière a été accordée à la résolution des bugs qui ont été identifiés lors du déclenchement du bouton espace dans la version précédente. Les anomalies qui pouvaient survenir lors de l'appui et du relâchement de la touche espace ont été méticuleusement corrigées. Ces ajustements visent à garantir une gestion fluide et fiable de la reconnaissance vocale, offrant ainsi une expérience utilisateur améliorée. Les corrections apportées contribuent à la stabilité globale de l'application, permettant aux utilisateurs d'initier et d'interrompre l'enregistrement vocal de manière cohérente et sans interruption.



4.4 Version 3:

Objectifs de la version 3 :

- ***Intégration de la Reconnaissance Vocale :***

Objectif : Mettre en œuvre un système de reconnaissance vocale capable de détecter une commande spécifique, telle que "Jarvis", lorsque prononcée par l'utilisateur.

Approche : Utiliser les technologies appropriées, comme l'API Web Speech, pour écouter en continu et déclencher l'analyse de la commande dès que le mot-clé spécifique est identifié.

- ***Enregistrement Audio :***

Objectif : Lorsque la commande vocale est détectée, arrêter l'écoute en cours et commencer l'enregistrement audio pour capturer la suite des instructions de l'utilisateur.

Approche : Implémenter une logique permettant de passer du mode d'écoute au mode d'enregistrement dès que la commande spécifique est reconnue.

- ***Transcription avec l'API Whisper :***

Objectif : Utiliser l'API Whisper d'OpenAI pour transcrire le fichier audio en texte, permettant ainsi de comprendre et d'interpréter les instructions de l'utilisateur.

Approche : Envoyer le fichier audio enregistré à l'API Whisper, récupérer la transcription textuelle résultante, et intégrer ces informations dans le flux de la conversation.

Fonctionnalités implémentées :

- ***Détection de la Commande "Jarvis" :***

La fonction utilise l'événement 'result' de l'objet recognition pour écouter les résultats de la reconnaissance vocale.

Elle parcourt les résultats actuels pour identifier si la phrase détectée contient le mot-clé "Jarvis".

La détection est basée sur la comparaison avec le résultat de la transcription après avoir été triée (élimination des espaces inutiles).

- **Réaction à la Commande Détectée :**

Si la phrase est considérée comme finale (isFinal), la fonction réagit à la détection de la commande "Jarvis".

Si la phrase comprend "Jarvis", elle change l'apparence du bouton de reconnaissance vocale (speechButton) en indiquant l'état d'écoute ou d'arrêt.

En cas de détection de la commande "Jarvis", elle arrête la reconnaissance vocale, déclenche la synthèse vocale pour répondre par "Oui", et commence l'enregistrement audio.

- **Gestion de l'Interface Utilisateur :**

La fonction met à jour l'apparence du bouton de reconnaissance vocale pour refléter l'état actuel, facilitant la compréhension visuelle de l'utilisateur.

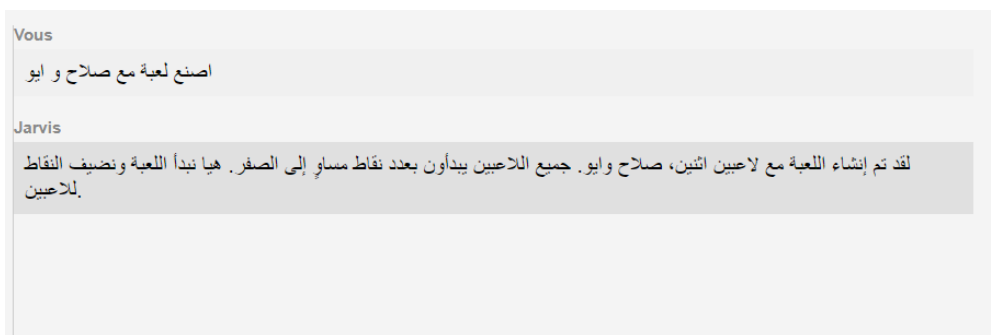
- **Cycle Continu de Reconnaissance Vocale :**

La fonction s'assure que la reconnaissance vocale continue après son achèvement, maintenant ainsi une écoute constante pour de nouvelles commandes.

Challenges rencontrés :

La reconnaissance vocale ne prend pas en compte la phrase pourtant mémorisée ("Il faut dire le mot-clé") et attendre 5 secondes avant de parler, ce qui peut entraîner des retards indésirables.

La reconnaissance vocale peut présenter des difficultés lorsqu'une personne parlant avec un accent ou dans une autre langue (par exemple, un locuteur arabophone parlant en français), provoquant des erreurs de transcription.



Solutions apportées :

Dans les prochaines versions, opter pour la reconnaissance vocale avec l'API Web Speech pour une détection directe des mots-clés. Cette approche pourrait simplifier le processus en éliminant le besoin d'attendre un délai spécifique avant de parler, offrant ainsi une expérience utilisateur plus réactive et intuitive.

Utiliser la reconnaissance vocale avec l'API Web Speech, qui permet la configuration explicite de la langue. Cette approche offre une flexibilité accrue pour ajuster les paramètres de langue en fonction des besoins spécifiques de l'application.

4.5 Version 4:

Objectifs de la version 4 :

- ***Implémentation de l'Historique de Conversation :***

Intégrer une fonctionnalité d'historique de conversation entre l'utilisateur et l'API GPT. Cela permettra de stocker et d'afficher les échanges précédents, offrant ainsi à l'utilisateur une vue contextuelle de la conversation en cours.

- ***Amélioration de la Détection du Mot-Clé "Jarvis" avec Web Speech :***

Optimiser la détection du mot-clé "Jarvis" lors de l'utilisation de la reconnaissance vocale avec l'API Web Speech. L'objectif est d'améliorer la réactivité et la fiabilité de la détection pour une interaction utilisateur plus fluide.

- ***Fonctions Calling :***

Implementes des fonctions qui permettent de supprimer un joueur , de recommencer un jeu, définir un score gagnant.et d'ajouter un joueur.

Fonctionnalités implémentées :

- ***Détection du Mot-Clé "Jarvis" avec Web Speech :***

La fonction Dit La Voix() utilise la reconnaissance vocale avec l'API Web Speech pour détecter le mot-clé "Jarvis" lors des interactions utilisateur.

Elle écoute les événements 'resultat' pour obtenir les résultats de la reconnaissance vocale.

En cas de détection d'une phrase finale avec une confiance suffisante et contenant le mot-clé "Jarvis", elle change l'apparence du bouton de reconnaissance vocale et extrait la commande vocale suivant le mot-clé.

La commande extraite est ensuite envoyée à la fonction `sendRequest()` pour traitement, et l'apparence du bouton est restaurée après l'envoi de la requête.

- *Affichage de l'Histoire de la Conversation :*

La fonction `update Conversation Display()` est responsable de mettre à jour l'interface utilisateur avec l'historique de la conversation entre l'utilisateur et Jarvis.

Elle prend en compte les messages échangés entre l'utilisateur et Jarvis, les organise par rôle (utilisateur ou assistant), et les affiche de manière chronologique.

L'historique de la conversation est stocké dans le tableau `conversationHistory` pour référence et suivi.

- *Implémentation de Fonctions Calling :*

Plusieurs fonctions calling ont été intégrées pour permettre une interaction plus avancée avec l'assistant GPT. Ces fonctions incluent :

Suppression d'un Joueur (`delete_player`) :

Permet à l'utilisateur de supprimer un joueur du jeu en spécifiant le nom du joueur à supprimer.

Répétition du Jeu (`repeat_game`) :

La fonction “`repeat_game`” permet à l'utilisateur de redémarrer le jeu à tout moment en réinitialisant les points des joueurs. L'utilisateur doit spécifier la liste des joueurs et les points associés.

Enregistrement du Score du Gagnant (winner_score) :

Permet à l'utilisateur de spécifier un score de victoire. Si l'un des joueurs atteint ce score, le jeu se termine et le gagnant est déterminé comme le joueur ayant le score le plus élevé. Il est interdit d'ajouter des points au joueur après avoir atteint le score défini.

Ajout d'un Nouveau Joueur (add_player) :

Permet à l'utilisateur d'ajouter un nouveau joueur au jeu en spécifiant le nom du joueur à ajouter.

Challenges rencontrés :

Le défi majeur a été la contrainte de la limite de MAX_HISTORY_LENGTH imposée par GPT, qui ne peut pas dépasser un certain nombre de tokens (environ 4700 tokens). Cela a entraîné des difficultés dans la gestion de l'historique de la conversation, en particulier lorsqu'il est nécessaire de stocker un grand nombre d'interactions.

Un autre challenge significatif a été l'observation que GPT ne suit pas toujours de manière cohérente les instructions fournies. Cela a conduit à des résultats imprévisibles dans les réponses générées par l'assistant, affectant la qualité globale de l'interaction.

Solutions apportées :

Pour surmonter la limitation de la longueur de l'historique MAX_HISTORY_LENGTH, une fonctionnalité a été ajoutée pour gérer dynamiquement la taille de l'historique. La constante MAX_HISTORY_LENGTH est définie à une valeur spécifique (40 dans cet exemple) pour limiter le nombre de messages conservés.

La fonction addToConversationHistory a été modifiée pour ajouter les nouveaux messages à l'historique et, en même temps, surveiller la taille totale de l'historique. Si la limite est dépassée, la fonction élimine automatiquement les messages les plus anciens, assurant ainsi que l'historique reste dans les limites spécifiées.

Cette solution a permis de maintenir une taille d'historique gérable, évitant ainsi les dépassements de limites tout en fournissant un contexte adéquat pour l'interaction avec l'assistant GPT. La mise à jour automatique de l'interface utilisateur avec la fonction `updateConversationDisplay` garantit une expérience utilisateur fluide.

4.6 Version 5

Objectifs de la version 5 :

- Reconnaissance immédiate de la phrase émise juste après le mot clef Jarvis
- Réalisation d'une interface simple et facile à utiliser
- Ajout de bouton pour des fonctionnalités rapide
- Test et Résolution

Les objectifs majeurs de la version 5 sont centrés sur l'amélioration de l'expérience utilisateur et la détection de bugs suite à plusieurs tests. Tout d'abord, nous visons à atteindre une reconnaissance immédiate de la phrase prononcée juste après le mot clé, permettant ainsi une interaction plus fluide et naturelle qui permettra de réduire le temps de réponse, renforçant ainsi l'efficacité globale de l'application.

En parallèle, nous nous concentrons sur la conception d'une interface utilisateur simple et facile à utiliser. L'objectif est de rendre l'interaction simple et accessible même pour ceux qui ne sont pas familiers avec la technologie vocale.

Dans le but d'optimiser les fonctionnalités offertes, la version 5 introduira des boutons dédiés pour des actions rapides. Ces boutons offrent une solution intuitive pour exécuter des commandes fréquemment utilisées, améliorant ainsi l'efficacité opérationnelle et offrant une expérience utilisateur plus personnalisée.

Enfin, Nous testerons et remonterons les bugs de la version 5 pour permettre au mieux une utilisation optimale et une interaction logique.

Fonctionnalités implémentées :

Les fonctionnalités implémentées dans cette version reposent principalement sur une base solide de HTML/CSS, offrant ainsi une interface utilisateur attrayante et intuitive. Cela nous permettra d'avoir une interface conviviale, favorisant une expérience utilisateur visuellement agréable et facile à naviguer. En ce qui concerne les fonctionnalités rapides, des conditions spécifiques tel que les boutons "Recommencer", "Choix de la vocalisation de Jarvis" et "Choix de la reconnaissance vocale" ont été incorporées pour permettre une expérience utilisateur personnalisée et adaptable. Ces conditions permettent une interaction rapide et sans heurts avec l'application, offrant aux utilisateurs la possibilité d'accomplir des tâches fréquemment utilisées de manière efficace.

Une attention particulière a été portée à l'amélioration de la fonction "Dit la voix" et à la détection des silences. Cette fonctionnalité a été optimisée pour une reconnaissance vocale plus précise et réactive, permettant une interaction plus naturelle avec l'assistant vocal Jarvis. La détection du blanc entre les phrases a été perfectionnée pour assurer une identification précise du début et de la fin de chaque commande vocale, contribuant ainsi à une expérience utilisateur homogène et sans interruption. Ces améliorations témoignent de notre engagement constant à fournir un système hautement performant et à répondre aux besoins évolutifs de nos utilisateurs.

Challenges rencontrés :

Des défis significatifs et certaines complexités ont été rencontrés au cours du développement, notamment en ce qui concerne la communication avec l'API. Malgré les instructions claires émises au niveau des "function calling", les réponses de l'API ne suivent pas toujours ces directives et peuvent parfois aller à l'encontre des attentes. Cette divergence peut entraîner des complications dans le traitement des données.

Un autre défi majeur réside dans la gestion des demandes récurrentes de l'utilisateur, particulièrement lorsque les requêtes deviennent fréquentes et similaires. Par exemple, si un utilisateur demande plusieurs fois l'ajout de points de manière similaire, le système ne fait plus appel à la fonction de rappel mais renvoie plutôt une réponse similaire à la requête similaire demandée par l'utilisateur précédemment. Ceci engendre des problèmes de mise à jour du

tableau car il ne prend plus en compte la fonction "calling" et n'ajoute pas les points même si Jarvis le dit.

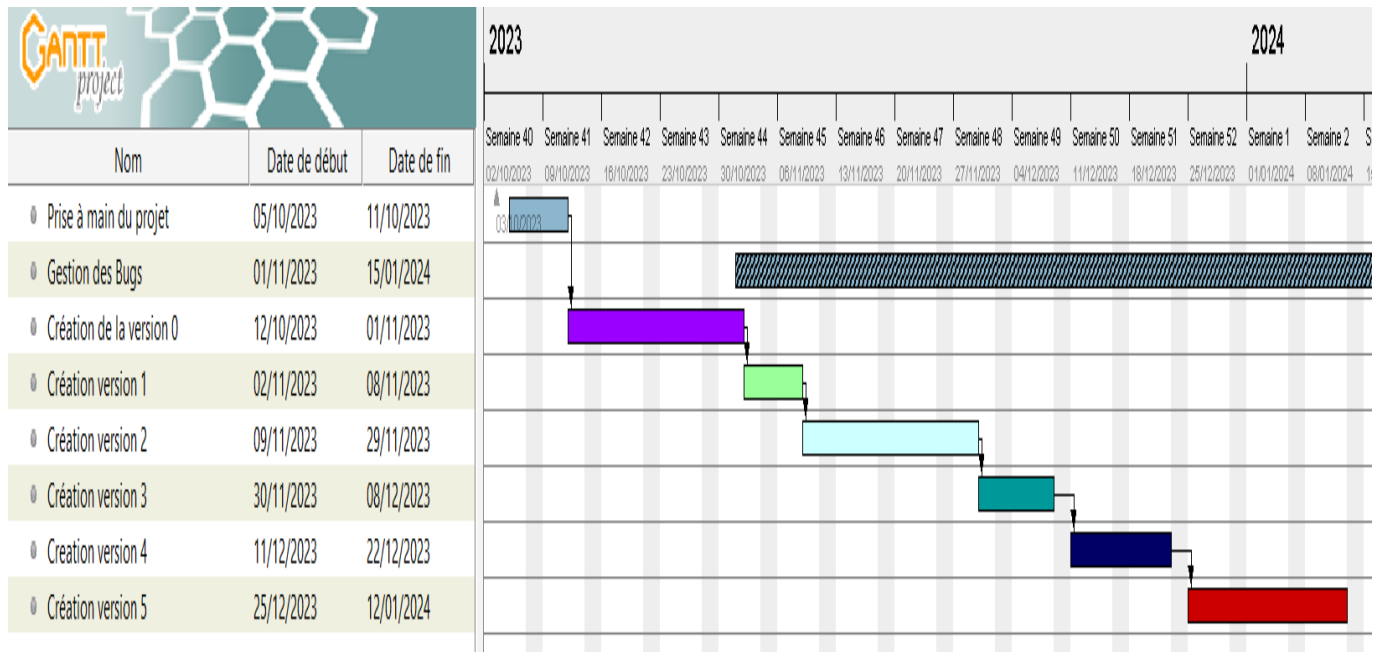
Face aux défis rencontrés, une solution a été apportée pour pallier aux bugs cités précédemment malgré l'apparition moins systématique de ces dernières . Pour remédier aux divergences dans les réponses de l'API malgré des instructions claires émises au "function calling", des efforts substantiels ont été déployés pour préciser davantage les instructions données à l'assistant et au système (exemple ci-dessous). Cette approche vise à réduire les bugs potentiels et à améliorer la cohérence des résultats obtenus.

```
addtoConversationHistory(["system", "As an AI assistant a game has been created, you must follow these guidelines for all the rest of the request:\n" +  
"- Always do not create an other game after creating the first game that have the list of players "+ getPlayers + " \n" +  
"- Only invoke the 'add_points' function for existing players in the list " + getPlayers + ".\n" +  
"- never add points without invoke the 'add_points' for players in the list " + getPlayers + ".\n" +  
"- Always reject the creation of a game or the invocation of the 'create_game' function .\n" +  
"- Only use functin calling to answer the user request \n. always invoke the 'repeat_game' function to repeat a game every time the user request \n ."+  
" Respond in French for all interactions.the players are "+ getPlayers + " there points successively are " + getpoints + " \n"]);  
addtoConversationHistory(["user", userInput]);
```

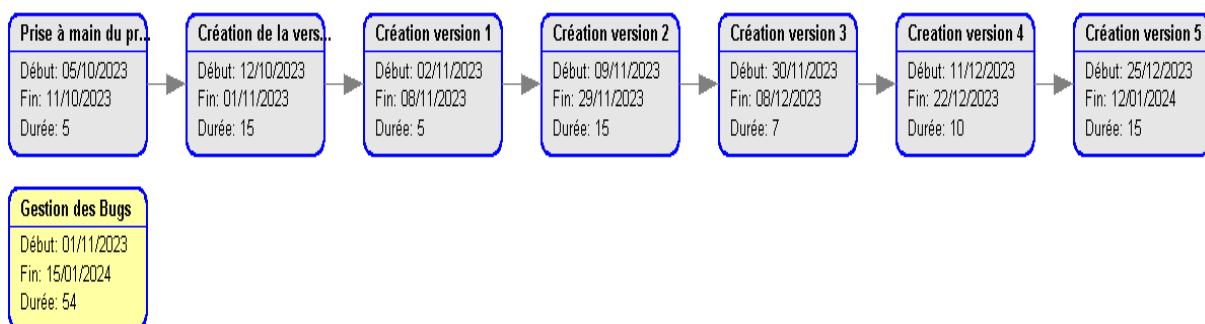
Cependant, malgré ces efforts, certaines limites à l'API ont été identifiées. Il est intéressant de noter que des problèmes similaires ont été soulevés par d'autres utilisateurs sur les forums d'OpenAI. Cette reconnaissance de problématiques partagées a conduit à des échanges constructifs sur le site, fournissant des pistes de solution futur et des perspectives pour améliorer le fonctionnement du système.

5-Gestion des tâches :

5-1: Diagramme de Gantt



5-2: Diagramme de Pert



6. Perspective et Amélioration :

- Interface Utilisateur Améliorée :

Ajoutez des animations ou des indicateurs visuels pour signaler l'état actuel du jeu, comme le démarrage ou l'arrêt de l'enregistrement vocal.

Affichez des messages plus explicites pour guider l'utilisateur tout au long du processus de jeu, en particulier lorsqu'il interagit avec Jarvis.

- Optimisation du Code :

Organisez le code en utilisant des fonctions modulaires et des commentaires pour améliorer la lisibilité et la maintenance.

Vérifiez la gestion des erreurs pour des interactions plus robustes avec l'API OpenAI, en fournissant des messages d'erreur explicites en cas de problème.

- Fonctionnalités de Reconnaissance Vocale :

Ajoutez une fonctionnalité pour activer et désactiver la reconnaissance vocale en appuyant sur une touche spécifique plutôt que d'utiliser une variable globale. Cela pourrait rendre l'expérience plus flexible.

Permettez à l'utilisateur de choisir la langue pour la reconnaissance vocale, offrant ainsi une expérience plus personnalisée.

- Fonctions du Jeu et Feedback Vocal :

Ajoutez des fonctionnalités supplémentaires au jeu, par exemple, des commandes spécifiques pour obtenir des informations sur le classement des joueurs, le score actuel, etc.

Intégrez davantage de feedback vocal pour informer l'utilisateur sur l'état du jeu, les résultats des actions, et les règles spécifiques.

- Gestion des Points et Fin de Jeu :

Améliorez la gestion des points et la détection de la fin de jeu. Assurez-vous que les règles du jeu sont strictement suivies et que le modèle informe clairement l'utilisateur du gagnant.

- Réactivité de l'Interface Utilisateur :

Assurer que l'interface utilisateur est réactive et offre une expérience utilisateur fluide, notamment en évitant des blocages ou des erreurs imprévus.

7.Conclusion :

Grâce à ce projet, nous avons établi une base solide dans le domaine de l'intelligence artificielle, en particulier des chatbots, en intégrant avec succès des technologies avancées telles que la reconnaissance vocale, la transcription audio et la synthèse vocale. Nous tenons à exprimer notre profonde gratitude envers les responsables et l'université de la Côte d'Azur pour nous avoir offert cette opportunité de développer nos compétences et de repousser les limites de l'innovation. Ce projet a ouvert de nouvelles perspectives prometteuses pour l'évolution des interfaces vocales interactives et a renforcé notre compréhension du fonctionnement des chatbots. Nous sommes fiers des succès techniques et conceptuels que nous avons atteints et nous sommes impatients de continuer à innover dans le domaine de l'intelligence artificielle conversationnelle.

8-Glossaire :

- **API (Interface de Programmation d'Application)** : Ensemble de règles et protocoles qui permettent à un logiciel de communiquer avec un autre.
- **Web Speech API** : API JavaScript permettant la reconnaissance vocale dans les applications web.
- **MediaRecorder** : Interface web pour enregistrer de l'audio ou de la vidéo directement dans le navigateur.
- **Transcription Audio** : Processus de conversion de l'audio parlé en texte écrit.
- **Synthèse Vocale (API SpeechSynthesis)** : Technologie de génération de la parole humaine à partir de texte.
- **Interface Utilisateur (UI)** : Ensemble des éléments visuels et interactifs avec lesquels l'utilisateur interagit dans une application.
- **Console de Débogage** : Interface interactive pour tester, déboguer et surveiller l'exécution d'un programme.
- **Clé API** : Code d'authentification permettant à une application d'accéder à des services externes.
- **Réponse Vocale (Speech Synthesis Utterance)** : Objet représentant le contenu vocal à synthétiser avec l'API SpeechSynthesis.
- **Historique du Jeu** : Ensemble d'enregistrements ou de données qui trace l'évolution d'une partie du jeu.
- **GPT (Generative Pre-trained Transformer)** : Pré-entraînement sur données textuelles pour générer du texte et effectuer des tâches de traitement du langage naturel.

9-WEBOGRAPHIE:

- <https://platform.openai.com/docs/guides/gpt>
- https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API/Using_the_Web_Speech_API
- <https://github.com/mdn/dom-examples/tree/main/web-speech-api>
- <https://platform.openai.com/docs/api-reference/audio>
- [dom-examples/web-speech-api at main · mdn/dom-examples](#)
- <https://www.commentcoder.com/api-chatgpt/>