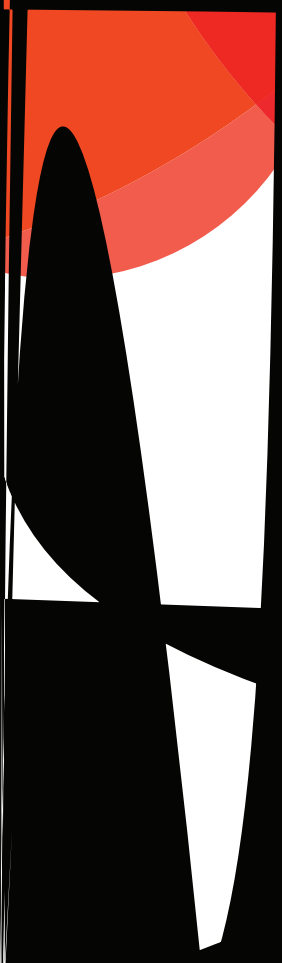
Rapport de Projet MemTogether

Aymane, Sunny, Mohamed, Salma

Mai 2025



Sommaire

I. Introduction

Le projet MemTogether est une plateforme web collaborative développée par une équipe de quatre étudiants dans le cadre du module AMS. Elle permet à plusieurs utilisateurs de se retrouver en ligne pour regarder ensemble des vidéos ou photos, discuter en temps réel, et réagir simultanément au contenu partagé.

Alternative enrichie à des plateformes comme Watch2Gether, MemTogether se distingue par une orientation familiale, interactive et conviviale.

Le développement du projet a suivi une approche en cascade : analyse, développement, tests, puis déploiement, avec une collaboration constante via GitHub et Discord.

II. Objectifs et périmètre fonctionnel

2.1 Objectifs

- Créer une plateforme web responsive, moderne et fluide
- Permettre le visionnage synchronisé de vidéos et photos
- Offrir un chat texte et vocal en temps réel
- Intégrer des fonctionnalités de réaction, suppression, et modération
- Autoriser l'utilisation de contenus YouTube ou personnels

2.2 Fonctionnalités clés (tout est synchronisé)

La plateforme MemTogether a été conçue pour offrir une expérience interactive complète, conforme aux spécifications fonctionnelles du cahier des charges. Elle intègre une série de fonctionnalités indispensables visant à recréer en ligne une ambiance de visionnage collectif, tout en assurant sécurité, fluidité et convivialité.

- Création et gestion de salons (rooms) : Chaque utilisateur connecté peut créer une room, lui attribuer un nom, puis partager un lien avec d'autres utilisateurs afin qu'ils rejoignent la même session.
- Authentification sécurisée : Système complet incluant la création de compte, la connexion sécurisée avec vérification des identifiants, et la gestion des sessions grâce à Django Auth.
- Synchronisation en temps réel : Grâce à l'implémentation des WebSockets via Django Channels, les utilisateurs d'un même salon voient le même contenu au même moment.
- Chat texte et vocal : Communication instantanée permettant aux membres du salon d'échanger pendant la session sans rechargement de page.
- Réactions en direct : Possibilité de réagir au contenu diffusé à l'aide d'émojis ou de messages visibles instantanément par tous les participants.
- Système de suppression : Les utilisateurs peuvent supprimer localement et dans le cloud certaines images ou vidéos partagées durant la session.
- Prévention des doublons : Fonction empêchant de mettre la même image deux fois.
- Mode visionnage optimisé : Permettant une meilleure qualité de visionnage.
- Système de modération : Fonction d'exclusion ou de promotion d'utilisateurs accessible uniquement au modérateur (créateur de la room).
- Contact support : Possibilité d'envoyer directement un mail au service client via la plateforme.
- Filtrage de contenus : Fonction permettant de ne pas partager des contenus obscènes.
- Apparition du modérateur en direct : Le modérateur peut apparaître en direct sur la room et parler, il est alors visible pour tous les utilisateurs présents.
- Modification du profil : Possibilité de modifier les informations du compte utilisateur via la page profil.

- Intégration YouTube : Visionnage direct des vidéos hébergées sur la plateforme via un player embarqué.
- Design responsive : Grâce à l'utilisation de Bootstrap, permettant une navigation confortable sur différents appareils.
- Mode invité : Permet à un utilisateur de rejoindre une room via l'identifiant en mode spectateur, avec un accès limité mais incluant le chat, les réactions et l'ajout d'images dans la file d'attente.
- Acceptation des CGU : Pour des raisons juridiques et de transparence, un système de pop-up bloquant impose l'acceptation des Conditions Générales d'Utilisation.

III. Démarche et Méthodologie

3.1 Planification (méthode en cascade)

Phase	Période	Résultats attendus
Analyse et conception	Semaines 1–3	Cahier des charges, architecture, Gantt
Développement initial	Semaines 4–6	Backend, WebSocket, Frontend
Tests et validation	Semaines 6–7	Tests unitaires et utilisateurs
Déploiement final	Semaine 8	Livraison, nettoyage du code

Table 1. Planification du projet selon la méthode en cascade

3.2 Répartition des rôles

- Backend : Aymane, Sunny, Mohammed
- Frontend : Salma
- Tests & déploiement : Salma
- Livrables : Tous

Même si comme mentionné précédemment, tous les membres ont été amenés à travailler sur le backend et le frontend.

Table 2. Répartition fidèle et détaillée des tâches

Nom	Durée	Tâche
Salma	3h	Recherche des technologies et frameworks
Aymane	2h	Recherche des technologies et frameworks
Mohamed	3h	Recherche des technologies et frameworks
Salma	2h	Recherche sur les fonctionnalités principales de Watch2Gether et fonctionnalités concu
Aymane	1h	Recherche sur les fonctionnalités principales de Watch2Gether et fonctionnalités concu
Mohamed	1h	Recherche sur les fonctionnalités principales de Watch2Gether
Aymane	20min	Recherche de fonctionnalités pertinentes à ajouter sur notre plateforme concurrente
Sunny	4h	Recherche sur le contenu du cahier des charges
Sunny	4h	Création du cahier des charges et intégrations des recherches préliminaires
Salma	3h	Renforcement des compétences en JavaScript et Python
Aymane	1h	Renforcement des compétences en JavaScript et Python
Aymane	30min	Recherche d'un framework pour le front
Mohamed	2h30	Bootstrap CSS pour templates des pages
Sunny	4h	Renforcement des compétences en programmation web
Salma	2h	Renforcement des compétences Bootstrap
Sunny	2h	Compréhension de Bootstrap et Django

Suite de la répartition

Nom	Durée	Tâche
Salma	5h	Révision et approfondissement du cahier des charges
Aymane	1h	Révision et approfondissement du cahier des charges
Salma	6h	Révision et approfondissement du cahier des charges
Mohamed	1h	Révision et approfondissement du cahier des charges
Aymane	2h	Création du projet Miro (diagramme Gantt)
Aymane	3h	Création de la base du projet (Django, authentification, architecture templates, GitHul
Mohamed	7h30	Adaptation d'un projet existant d'authentification
Sunny	4h	Début de l'implémentation des vidéos YouTube et compréhension des requêtes
Salma	6h	Développement des premiers designs de la page d'accueil et de connexion
Aymane	40min	Mise à jour du diagramme Gantt et redistribution des tâches
Salma	2h	Préparation Bot (extraction d'image avec Google Drive)
Aymane	7h	WebSocket et chargement des images dans les sessions depuis un dossier local (23-26 m
Salma	6h	Gestion des utilisateurs d'une session (liste, déconnexion, modération, exclusion) (25-29
Mohamed	30min	Mise à jour diagramme de Gantt
Mohamed	6h	Implémentation upload multi-images avec backend Django, Cloudinary et WebSocket
Salma	4h	Gestion des bugs session utilisateur (5-6 avril)
Sunny	12h	Implémentation vidéos YouTube (BE+FE), tchat et correction bugs WebSocket (1-2 et
Mohamed	9h	supression des utilisateurs,Amelioration backend modification du websocket
Mohamed	4h	Navigation manuelle des images,selection image fils d'attente
Mohamed	5h	gestion déconnexions automatiques,ajout du systeme de réactions
tous	2h	Création d'une charte graphique (12 avril)
Aymane	10h	Chargement des images depuis le cloud et file d'attente (10-12 avril)
Aymane	8h	Front accueil/login épuré, correction redirections, mdp salons chiffrés (13-14 avril)
Aymane	4h	Front room épuré et switch entre options YouTube/photos (15 avril)
Aymane	2h	Correction bugs connexion et upload images (16 avril)
Salma	3h	Backend modification profil après inscription (14 avril)
Salma	3h	Gestion des bugs des pages (16 avril)
Aymane	30min	Mise à jour du diagramme de Gantt (16 avril)
Équipe	20min	Réunion avancement et partage des tâches (16 avril)
Sunny	6h	Correction bugs, amélioration frontend et finalisation tchat (16 avril)
Sunny	4h	Recherche SimplePeer pour système cam/audio et début implémentation (19 avril)
Sunny	10h	Ajout messages d'erreur, débuggage et finalisation stream BE+FE (22-23 avril)
Salma	5h	Création CGU avec système d'acceptation obligatoire (20-22 avril)
Salma	2h	Gestion bugs interactions temps réel (smileys, chat) (21 avril)
Salma	2h	Front (22 avril)
Salma	1h	Front page CGU (26 avril)
Salma	1h	Front page profil (28 avril)
Sunny	6h	Test du tchat et facecam + tentatives de crash (27 avril et 2 mai)
Sunny	1h	Intégration mode visionnage et plein écran (29 avril)
Mohamed	3h	Suppression des images dans la file d'attente (29 avril)
Aymane	5h	Empêcher images inappropriées avec RapidAPI et boutons pause/play (29 avril)
Mohamed	1h30	Correction bugs apparus lors de la suppression (1 mai)
Mohamed	3h30	Fonction anti-doublons avec hashage (1 mai)
Mohamed	2h	Affichage de l'auteur des images (3 mai)
Aymane	4h	Suppression images du Cloud en fin de session et depuis file d'attente (2 mai)
Salma	1h	Front end de la room (1 mai)
Salma	1h	Front end de la page profil (2 mai)
Salma	5h	Ajout page contact backend avec serveur SMTP pour mail au service client (3-5 mai)
Sunny	4h	Test fonctionnel et unitaire sur les fonctionnalites essentielles

Suite de la répartition

Nom	Durée	Tâche
Salma	1h	Réarrangement des paramètres et réglage bugs (4 mai)
Salma	3h	Tests complets (photos multiples, vidéos, durée 1h, suppression photos, caméra, audio)
Mohamed	3h	Tests complets (photos multiples, vidéos, durée 1h, suppression photos, caméra, audio)
tous	en-cour	Rédaction du rapport

3.3 Liste des tâches personnelles

Travail réalisé par Mohamed Adnane

Au cours du projet MemTogether, jai principalement contribué aux aspects techniques du backend, à la gestion des WebSockets, ainsi quà lintégration de plusieurs fonctionnalités complexes.

Dans un premier temps, jai participé à la recherche des technologies et frameworks à utiliser pour le projet, ce qui ma permis de bien cerner les besoins techniques liés à Django et à la communication en temps réel.

Jai ensuite assuré la création et ladaptation dun projet dauthentification existant, que jai entièrement intégré et modifié pour correspondre à larchitecture de notre application. Cette tâche ma pris plus de 7 heures et a constitué un point de départ crucial pour la structuration du backend.

Par la suite, jai travaillé sur limplémentation de lupload multi-images côté client, leur envoi vers le backend via requêtes asynchrones, et leur stockage sur Cloudinary, tout en assurant la mise à jour en temps réel via WebSocket.

Jai également consacré beaucoup de temps au développement de fonctionnalités avancées, notamment la gestion de la navigation manuelle des images (précédent, suivant, clic direct dans la file), la détection des utilisateurs inactifs, leur suppression automatique de la session, et le transfert du statut de créateur si ce dernier quitte la room. Jai aussi implémenté un système de réactions en temps réel avec emojis, visible par tous les participants.

En parallèle, jai pris en charge la correction de plusieurs bugs liés aux suppressions dimages, notamment lors de limplémentation de la suppression directe depuis la file dattente, et jai ajouté une fonction anti-doublons basée sur un système de hachage afin dempêcher lajout multiple dune même image. Jai également mis en place la possibilité de voir quel utilisateur a ajouté quelle image, pour plus de transparence dans les sessions.

Enfin, jai participé activement à la phase de tests finaux, en vérifiant le bon fonctionnement des différentes fonctionnalités : ajout/suppression dimages, affichage synchrone, gestion multi-utilisateurs, et compatibilité avec les sessions longues. Jai également contribué à la rédaction du rapport, en relisant et structurant les parties techniques liées au backend et à la logique applicative.

Problèmes rencontrés : lintégration fluide de Cloudinary avec Django via les WebSockets a demandé plusieurs essais et ajustements, notamment pour synchroniser correctement les suppressions à distance. De plus, la détection fiable des utilisateurs inactifs et la gestion des déconnexions imprévues ont nécessité une adaptation fine du modèle et du consumer WebSocket, afin déviter les incohérences entre la base et laffichage en direct.

Travail réalisé par Aymane

Dans le cadre du projet MemTogether, jai joué un rôle polyvalent, en minvestissant aussi bien dans les tâches de recherche initiales que dans limplémentation concrète du backend, du frontend, et des fonctionnalités liées aux sessions.

Jai commencé par contribuer à la recherche des technologies et frameworks à utiliser, ainsi quà la recherche sur les fonctionnalités des plateformes concurrentes, ce qui nous a permis de définir une orientation technique cohérente et des fonctionnalités différenciantes pour notre projet.

Jai ensuite participé à la rédaction et à lamélioration du cahier des charges, ainsi quà la création du diagramme de Gantt sur Miro, qui a servi de base à lorganisation du projet. À partir de là, jai pris en charge la création de la base du projet Django, avec linitialisation des applications principales comme lauthentification et les salons, ainsi que la préparation du dépôt GitHub.

Par la suite, jai travaillé sur des fonctionnalités clés telles que la mise en place des WebSockets, le chargement des images dans les sessions, la création dune file dattente dimages, et lintégration avec Cloudinary. Cela a impliqué lécriture de logique côté serveur, mais aussi la synchronisation en temps réel des contenus sur les différents écrans.

Jai également assuré lintégration du front-end, en développant une interface fluide et réactive pour la page daccueil, la page de connexion et linterface principale des salons. Jai intégré des fonctionnalités comme la protection par mot de passe des salons, avec chiffrement/déchiffrement, et le changement dynamique de mode (YouTube ou images) durant une session.

En complément, jai travaillé sur la modération des images, notamment via lintégration de RapidAPI pour bloquer les contenus inappropriés, ainsi que sur lajout de contrôles vidéo comme les boutons pause/play. Jai aussi participé à la suppression des images du Cloud en fin de session ou manuellement depuis la file.

Enfin, jai effectué plusieurs tests fonctionnels, en simulant des cas réels dusage sur plusieurs navigateurs et utilisateurs, et jai participé à la réunion de coordination pour assurer une bonne répartition des tâches et vérifier l'avancement général.

Problèmes rencontrés : lune des principales difficultés a été de garantir une synchronisation fluide des images et vidéos entre tous les utilisateurs via les WebSockets, tout en maintenant de bonnes performances. Le filtrage via API a également nécessité un ajustement précis pour éviter les faux positifs. Par ailleurs, lintégration simultanée de plusieurs modules (authentification, média, modération) a demandé une attention particulière à la cohérence des routes, des modèles et de la logique métier.

Travail réalisé par Sunny

Dans le projet MemTogether, jai principalement été responsable de la partie gestion des flux médias, de lintégration des WebSockets, ainsi que du système de communication au sein des rooms.

Jai commencé par effectuer une recherche approfondie sur le contenu du cahier des charges, avant den assurer une partie de la rédaction et structuration initiale. Par la suite, je me suis formé en autonomie afin de renforcer mes compétences en programmation web, notamment avec JavaScript et Django, pour pouvoir intervenir efficacement dans les différentes couches du projet.

Lune de mes principales contributions a été limplémentation des vidéos YouTube, à la fois côté backend et frontend, ainsi que la création du système de tchat en temps réel. Jai également travaillé sur la correction de bugs liés aux WebSockets et aux synchronisations entre utilisateurs.

Jai ensuite mis en place un système de streaming vidéo et caméra grâce à la bibliothèque SimplePeer, en assurant aussi bien la configuration backend que le comportement frontend. Jai ajouté des messages derreur dynamiques, des tests de stabilité, et corrigé les problèmes liés aux connexions multiples ou aux changements de mode.

Jai aussi réalisé des tests fonctionnels approfondis, y compris des tentatives de crash, pour valider la robustesse du système. Jai finalisé la partie frontend du tchat, intégré le mode visionnage plein écran, et participé à la création de la charte graphique avec le reste de léquipe.

Problèmes rencontrés : La configuration de SimplePeer a été complexe, notamment pour gérer la négociation des flux entre pairs et la compatibilité navigateur. Jai aussi rencontré des conflits dévénements WebSocket entre la vidéo et le chat, qui ont nécessité une refactorisation du consumer. Le test simultané de la vidéo, du chat et des réactions ma demandé de nombreux ajustements pour garantir une expérience fluide et réactive.

Travail réalisé par Salma

Tout au long du projet MemTogether, jai assuré un rôle central dans la rédaction du cahier des charges, le design des interfaces, la gestion utilisateur, ainsi que lamélioration continue du frontend.

Jai commencé par contribuer à la recherche des technologies et des fonctionnalités des plateformes concurrentes, avant de participer activement à la rédaction du cahier des charges, que jai relu, corrigé et enrichi sur plusieurs sessions.

Par la suite, jai pris en charge la création des maquettes et des premiers designs de la page daccueil et de connexion. Jai réalisé plusieurs versions de linterface et collaboré avec les membres du groupe pour harmoniser lergonomie. Jai aussi participé à la mise en place de la charte graphique afin de rendre le projet cohérent visuellement.

Sur le plan technique, jai travaillé sur la gestion des utilisateurs dans les salons (affichage, déconnexion, modération, transfert de créateur), ainsi que sur la préparation dun bot dextraction dimage via Google Drive. Jai également implémenté un formulaire de modification de profil après inscription, et conçu les pages CGU, profil et contact.

Jai aussi développé une fonctionnalité backend complète pour le service client de la plateforme, via une page de contact permettant aux utilisateurs denvoyer des messages. Cela a nécessité la configuration dun serveur SMTP sécurisé, lécriture dune logique denvoi de-mails via Django, ainsi que la gestion des erreurs et des retours utilisateur pour confirmer la bonne transmission du message à notre adresse de support.

Enfin, jai été impliquée dans la gestion des bugs liés aux interactions, notamment avec le chat et les pages front. Jai également réalisé plusieurs tests utilisateurs, participé à la rédaction finale du rapport, et assuré des réunions de suivi davancement avec les autres membres de léquipe.

Problèmes rencontrés : Le principal défi a été de maintenir une cohérence graphique malgré les nombreux changements fonctionnels apportés au fil du temps. Jai également dû gérer les conflits entre les modifications frontend de chacun, ainsi que des soucis dinteractions dynamiques (affichage conditionnel, erreurs de redirection). Enfin, le lien entre le backend Django et les retours utilisateur dans linterface ma demandé des ajustements pour que tout soit fluide et intuitif.

3.4 Diagramme de Gantt

Le diagramme ci-dessous illustre lavancement du projet MemTogether semaine par semaine, en détaillant les contributions de chaque membre du groupe. Chaque couleur représente un membre,

et les barres indiquent la durée et le type de tâche effectué.

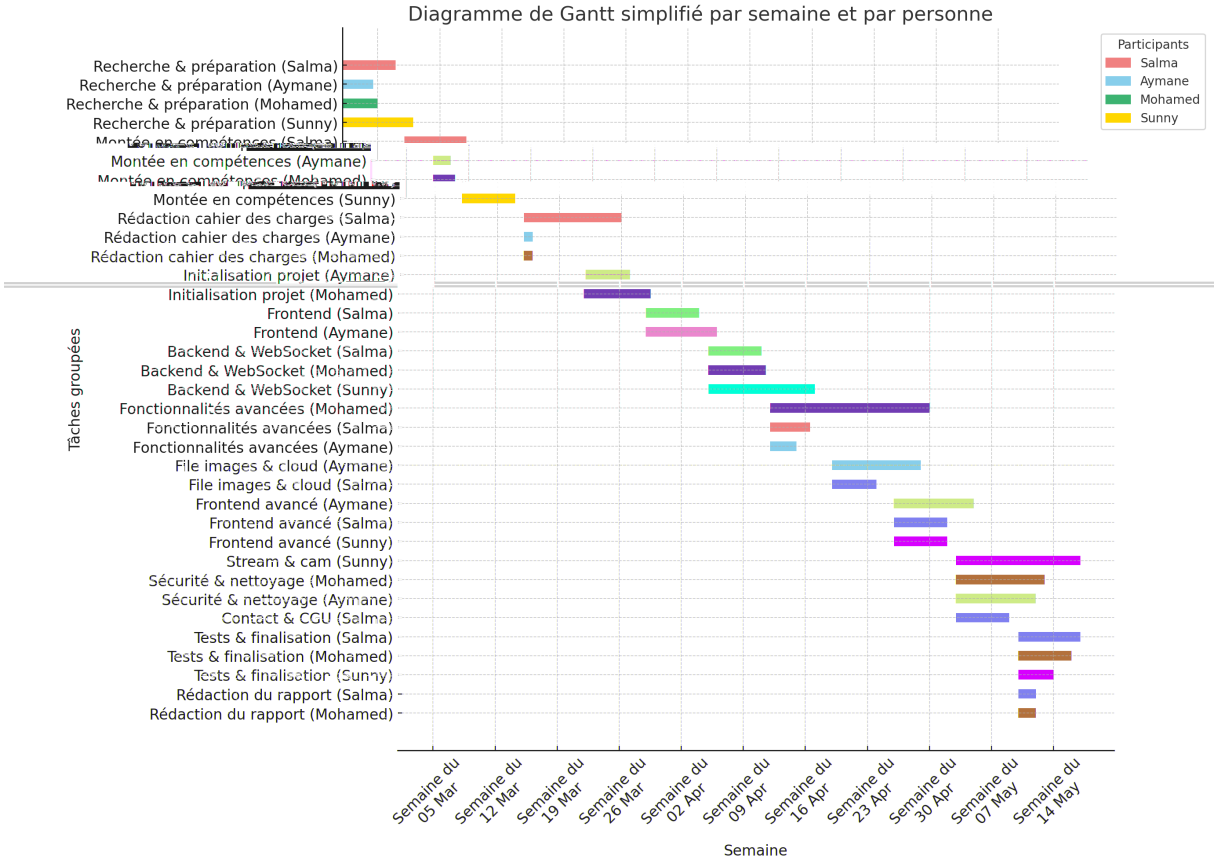


Figure 1. Diagramme de Gantt du projet MemTogether

IV. Architecture et Technologies

Le projet MemTogether repose sur une architecture web moderne, découpée en deux grandes parties : le backend, qui gère la logique métier et la communication en temps réel, et le frontend, qui offre une interface fluide et intuitive à l'utilisateur. Le choix des technologies s'est appuyé sur des critères de simplicité, d'efficacité et de compatibilité avec les fonctionnalités collaboratives et temps réel attendues.

4.1 Backend

Le backend du projet est développé en Python 3.11, un langage moderne, lisible et très utilisé pour le développement web. Le framework principal utilisé est Django, reconnu pour sa robustesse, sa rapidité de développement et son système intégré de gestion des utilisateurs, des bases de données et des formulaires.

Pour les fonctionnalités de communication en temps réel, notamment le chat, la synchronisation des vidéos et les réactions instantanées, le projet s'appuie sur Django Channels. Ce module étend les capacités de Django classiques (HTTP) aux protocoles asynchrones comme WebSocket, ce qui permet une communication bidirectionnelle entre le serveur et le navigateur. Les fichiers `asgi.py` et `routing.py` définissent l'infrastructure nécessaire à cette communication asynchrone.

La base de données utilisée est SQLite, un système de gestion léger, embarqué et suffisant pour le développement local ou les projets à petite échelle. Elle permet de stocker toutes les

entités nécessaires au fonctionnement du site : utilisateurs, rooms, contenus multimédias, votes, messages, etc.

4.2 Frontend

Le frontend repose sur un ensemble de technologies standards du web, combinées de manière cohérente pour assurer une interface moderne, responsive et interactive :

- Le HTML est utilisé pour structurer les différentes pages de l'application (connexion, accueil, room, galerie, etc.).
- Le CSS, enrichi par le framework Bootstrap, permet de styliser les pages avec un design professionnel tout en assurant une adaptabilité aux différentes tailles d'écran (responsive design).
- Le JavaScript est utilisé pour rendre l'interface dynamique : gestion des réactions en direct, envoi de messages sans rechargement de page, affichage conditionnel de certains éléments, etc.
- Enfin, les templates Django jouent un rôle central dans l'intégration dynamique des données entre le backend et le frontend.

4.3 Outils de collaboration

La réussite de ce projet s'appuie aussi sur une organisation rigoureuse et l'utilisation d'outils collaboratifs efficaces :

- GitHub : utilisé pour le versionnage du code, le suivi des contributions de chaque membre, et la gestion des branches et fusions.
- Discord : principal canal de communication de l'équipe, avec des salons dédiés pour discuter en temps réel, résoudre des blocages techniques et organiser les séances de travail.
- Framapad : utilisé pour l'avancement du projet et la rédaction collaborative.
- Miro : utilisé pour la création du diagramme de Gantt permettant de planifier les différentes phases du projet.

V. Développement Réel et Fonctionnalités Implémentées

Le développement du projet MemTogether a une attention particulière portée à l'implémentation progressive des fonctionnalités attendues.

5.1 Authentification et CGU

Un système d'authentification sécurisé a été mis en place afin de gérer les inscriptions et connexions des utilisateurs. Un formulaire personnalisé permet aux utilisateurs de créer un compte avec nom, mot de passe et autres informations, via un système basé sur le module `forms.py`. La vérification des identifiants et la gestion des connexions sont gérées dans `views.py` du module authentification.

Conformément aux exigences légales et fonctionnelles, une acceptation obligatoire des Conditions Générales d'Utilisation (CGU) est intégrée. Lors de la première connexion, un pop-up bloque l'accès à la plateforme tant que l'utilisateur n'a pas explicitement validé les CGU.

5.2 Rooms et synchronisation

La création de rooms dynamiques est l'une des fonctionnalités centrales de MemTogether, où un identifiant unique est généré pour chaque salon. Ces rooms servent de points de rassemblement pour les utilisateurs afin de partager des souvenirs ensemble.

La synchronisation en temps réel entre les utilisateurs d'une même room est assurée via Django Channels. Grâce aux WebSockets, toute action (comme le changement de média, une

réaction ou un message) est immédiatement répercutée chez tous les utilisateurs connectés à la room, assurant ainsi une expérience collaborative fluide et cohérente.

Les utilisateurs peuvent également basculer entre deux modes de session mode photo et mode YouTube et ce choix est synchronisé pour tous les participants. Si le modérateur change le mode, tous les utilisateurs basculent automatiquement vers le même type de contenu.

5.3 Lecture et partage multimédia

La plateforme permet la lecture et le partage de contenus multimédias, notamment des images et vidéos. Le modèle `ImageQueue` permet de stocker ces fichiers dans une file d'attente associée à chaque room. Les utilisateurs peuvent ainsi uploader des contenus, qui seront ensuite affichés dans l'ordre défini.

Plusieurs améliorations ont été intégrées :

- la possibilité de supprimer une image de la file dattente (pour tous les utilisateurs),
- la possibilité de sélectionner directement une image dans la file pour la visionner immédiatement sur tous les écrans,
- un système de filtrage via une API externe (RapidAPI) afin de refuser les images potentiellement inappropriées.

Le visionnage de vidéos YouTube est rendu possible grâce à l'intégration de lecteurs via HTML et scripts JavaScript, assurant une compatibilité avec le système de synchronisation.

5.4 Communication

La plateforme met en uvre un chat texte en temps réel, développé à l'aide de WebSockets dans le fichier `consumers.py`. Ce chat permet aux membres d'une même room d'échanger pendant la session, sans avoir à recharger la page.

Un système de visio (caméra et audio) a été intégré spécifiquement pour le modérateur de la room, afin de renforcer son rôle central dans la session et favoriser une meilleure animation du groupe.

5.5 Réactions

L'interface utilisateur propose des réactions en direct, sous forme d'émojis ou de likes. Ces réactions sont transmises en temps réel via WebSockets à tous les utilisateurs de la room, ce qui améliore l'interactivité et la convivialité des sessions.

5.6 Tests et robustesse

Concernant les tests, un fichier `test_email.py` a été mis en place pour tester l'envoi d'e-mails, notamment dans le cadre du formulaire de contact.

Chaque application (authentification, rooms, etc.) contient également un fichier `tests.py` destiné à valider certaines fonctionnalités spécifiques comme les modèles ou les vues. Les tests unitaires ont permis d'identifier rapidement des erreurs pendant le développement.

En complément, de nombreux tests manuels ont été réalisés en fin de projet pour s'assurer de la stabilité du système, notamment lors des démonstrations entre membres de l'équipe. Le débogage final a permis d'optimiser certaines parties du code et de corriger les incohérences restantes avant la livraison.

VI. Contraintes et Problèmes Rencontrés

Au cours du développement de la plateforme MemTogether, l'équipe a été confrontée à plusieurs contraintes techniques et organisationnelles. Bien que la majorité des fonctionnalités aient été

implémentées avec succès, certaines difficultés ont nécessité des ajustements, des compromis ou des solutions temporaires.

- Synchronisation vidéo complexe : L'une des premières difficultés techniques a été la gestion de la synchronisation des vidéos entre les utilisateurs d'une même room. Pour répondre à cette exigence, l'équipe a intégré Django Channels avec WebSockets, permettant une communication bidirectionnelle asynchrone entre le serveur et les navigateurs des utilisateurs.
- Limites de l'API YouTube : Un autre défi technique a concerné l'intégration de vidéos YouTube dans la plateforme. Initialement, certains essais de synchronisation via l'API YouTube rencontraient des problèmes de latence ou de contrôle restreint. Pour contourner ces limitations, l'équipe a opté pour l'intégration d'un lecteur YouTube via une balise iframe HTML, couplée à des scripts JavaScript.
- Absence de tests utilisateurs externes : Sur le plan méthodologique, une contrainte notable a été l'absence de tests utilisateurs externes. Pour pallier cela, des tests manuels ont été réalisés par les membres de l'équipe eux-mêmes, en se connectant via plusieurs comptes pour simuler différentes interactions.
- Sécurité : Concernant la sécurité des comptes utilisateurs, l'équipe a anticipé les risques en s'appuyant sur les mécanismes fournis par Django Auth, enrichis par un système de tokens, sessions et formulaires sécurisés. Les mots de passe sont stockés de manière chiffrée et aucune donnée critique n'est exposée dans le code.

En résumé, malgré plusieurs défis techniques et organisationnels, l'équipe a su faire preuve d'adaptabilité pour trouver des solutions efficaces, garantissant ainsi la cohérence du projet avec les attentes du cahier des charges.

VII. Réflexion sur la méthodologie

7.1 Limites de l'approche en cascade utilisée

Nous avons adopté une méthode de gestion en cascade (Waterfall) pour le développement de MemTogether. Cette approche linéaire, consistant à avancer étape par étape (analyse, conception, développement, test), nous a permis de poser un cadre clair dès le départ. Toutefois, elle a également montré ses limites dans un projet nécessitant des ajustements fréquents :

- Rigidité dans les évolutions : Lajout tardif de certaines fonctionnalités (caméra, filtrage dimages, suppression dans la file dattente) a nécessité de revenir sur des étapes déjà validées.
- Manque de tests intermédiaires : Lenchaînement linéaire des étapes dans la méthode en cascade ne prévoyait pas de phases de tests fréquentes, ce qui a parfois retardé la détection de petits bugs ou incohérences daffichage Mais on a quand meme fait des tests réguliers pour éviter le maximum d'erreurs.

7.2 Intérêts potentiels dune approche agile (Scrum)

Si nous avions opté pour Scrum, cela aurait pu présenter plusieurs avantages, notamment :

- Plus de flexibilité : Le fonctionnement par sprints aurait permis dintégrer plus facilement les fonctionnalités émergentes au fil du projet.
- Suivi davancement progressif : Grâce à des réunions régulières et des revues de sprint, nous aurions pu ajuster la direction technique de manière plus fine.
- Découpage fonctionnel efficace : Des itérations ciblées sur des blocs de fonctionnalités (upload, chat, WebSocket, etc.) auraient rendu le développement plus modulaire.

7.3 Une organisation réussie malgré tout

Malgré les limites inhérentes à lapproche en cascade, notre équipe a su faire preuve dune grande capacité dadaptation. Grâce à une communication régulière (Discord, réunions hebdomadaires)

et une cohésion déquipe solide, chacun a pu remplir ses tâches de manière équilibrée et autonome. La répartition du travail sest faite naturellement selon les compétences, et les jalons ont été respectés sans retard majeur.

Cette bonne dynamique a permis dabsorber les imprévus techniques tout en livrant une plateforme fonctionnelle et fidèle aux attentes du cahier des charges.

7.4 Vers une approche hybride

En rétrospective, une approche hybride aurait probablement été la plus adaptée à notre projet. Une planification initiale rigoureuse issue du modèle Waterfall aurait pu être combinée à des cycles de développement courts inspirés de Scrum. Cela aurait permis davoir une vision structurée du projet tout en restant flexible face aux contraintes techniques rencontrées et aux idées émergentes.

Cette réflexion nous amène à envisager, pour de futurs projets, un mode de gestion mixte alliant rigueur et adaptabilité.

7.5 Autocritique

Le développement de MemTogether nous a permis dacquérir de nombreuses compétences, mais également de prendre conscience de certaines de nos limites et erreurs :

- Organisation du dépôt Git : Parfois, des conflits ou des confusions dans les branches ont ralenti le développement. Une convention de nommage et une gestion plus stricte des merges auraient été bénéfiques.
- Dépendance à certaines technologies : Le choix dutiliser Django Channels et WebSockets sans expérience préalable a représenté un défi important. Une phase de prototypage précoce aurait été utile.
- Documentation technique incomplète : Certaines parties du projet (notamment la configuration WebSocket ou la file d'attente multimédia) auraient mérité une documentation détaillée pour faciliter la relecture ou la reprise du projet.

Ces remarques constituent une base de réflexion pour améliorer notre façon de travailler dans de futurs projets, notamment en renforçant la planification, la communication.

VIII. Résultats Obtenus

À l'issue du développement, la plateforme MemTogether atteint un niveau de réalisation très satisfaisant, conforme à plus de 99% des attentes formulées dans le cahier des charges initial. Le projet a été mené à terme dans les délais impartis, avec une couverture fonctionnelle presque complète, une interface conviviale, et une architecture technique stable.

- Plateforme fonctionnelle à 99% conforme au cahier des charges
- Interface responsive (Bootstrap), fluide et accessible sur différents appareils
- Fonctionnalités principales disponibles et stables
- Expérience utilisateur intuitive
- Seul point à finaliser : déploiement en ligne

La plateforme est opérationnelle sur l'ensemble de ses fonctionnalités principales. Les utilisateurs peuvent se créer un compte, se connecter, accéder à leur espace personnel, créer des salons de visionnage, partager et visionner des contenus multimédias, discuter en temps réel via un chat textuel, réagir aux vidéos, supprimer les éléments partagés... Chaque fonctionnalité a été développée de façon cohérente et intégrée dans un ensemble fluide et intuitif.

IX. Conclusion

Le projet MemTogether constitue une réussite à plusieurs niveaux, tant sur le plan technique que sur le plan organisationnel. En s'appuyant sur un cahier des charges clair et ambitieux, l'équipe a su concevoir et développer une plateforme web collaborative, moderne, interactive et parfaitement adaptée à l'objectif initial : permettre à des utilisateurs, notamment des familles, de se retrouver virtuellement pour partager des souvenirs multimédias de manière synchrone, comme s'ils étaient réunis dans la même pièce.

Les fonctionnalités majeures attendues ont été intégralement développées et fonctionnent de manière fluide : création de rooms, synchronisation des vidéos et images, chat texte en temps réel, votes collaboratifs, téléchargement de contenus, intégration YouTube, et acceptation des CGU. L'interface responsive et le design épuré assurent une expérience utilisateur agréable, cohérente et accessible depuis tout type de terminal.

L'architecture technique du projet repose sur une base solide, combinant Django pour la logique serveur, Django Channels pour la communication en temps réel via WebSockets, Bootstrap pour le frontend responsive, et SQLite pour la gestion de la base de données. Ce choix technologique a permis à l'équipe de construire un produit fiable, scalable et bien structuré.

Sur le plan humain, le projet a été une excellente opportunité pour renforcer les compétences transversales des membres de l'équipe. Il a mobilisé des connaissances en programmation web fullstack, en gestion de versions (Git/GitHub), en travail d'équipe, ainsi qu'en organisation en cascade du développement. Chaque membre a pu explorer plusieurs rôles (frontend, backend, tests, documentation), favorisant une polyvalence technique et une collaboration efficace.

En conclusion, MemTogether est un projet techniquement maîtrisé, fonctionnel et prometteur, qui peut désormais évoluer vers une version pleinement accessible en ligne. Il incarne l'esprit d'innovation, de rigueur et de coopération que requiert tout développement logiciel moderne. Ce travail constitue ainsi une base solide pour un éventuel déploiement professionnel, voire une extension future du projet avec des fonctionnalités supplémentaires.