CAHIER DES CHARGES

Projet IL 2025/2026

Plateforme collaborative de visionnage vidéo

17/11/2025

Prestataires

Taleb Wissame - Takdjerad Meriem - Ghabi Malek - Laftimi Yanis

Clients

ESSLER Remy - BONNEFOY Ludovic

Sommaire

1. Introduction

1.1 Objet du document

Le présent **cahier des charges** a pour objectif de définir de manière claire et structurée les besoins, les fonctionnalités et les contraintes liés au développement du projet **"With U"**, réalisé dans le cadre du module AMS Projet **Ingénierie Logicielle (IL)** du Semestre 5 à l'Université d'Avignon – CERI. Il sert de **référence commune** à l'équipe de développement et à l'enseignant encadrant, jouant le rôle de client. Le document précise les **services à fournir**, les **exigences techniques**, les **livrables attendus** ainsi que la **méthodologie de travail** adoptée. Il garantit ainsi une vision partagée du projet avant le lancement de la phase de conception et de développement.

1.2 Contexte et justification du projet

Dans un contexte où les interactions numériques et le travail collaboratif occupent une place centrale, de plus en plus d'utilisateurs souhaitent **partager à distance une même expérience audiovisuelle**, que ce soit pour se divertir, apprendre ou échanger.

Des plateformes comme *Watch2Gether* ou *Teleparty* répondent déjà partiellement à ce besoin, mais elles présentent des limites en matière d'ergonomie et de personnalisation.

Le projet **With U** s'inscrit dans cette dynamique : il vise à offrir une **solution simple, interactive et pédagogique** permettant à plusieurs utilisateurs de **regarder une vidéo en simultané** tout en **communiquant par un chat en temps réel**.

Ce projet constitue également un **exercice d'application des compétences** acquises en formation, notamment la gestion de projet, le développement web full stack et la conception d'applications collaboratives.

1.3 Objectifs du projet

Le projet **With U** poursuit à la fois des objectifs **pédagogiques** et **techniques**.

- ***Objectifs pédagogiques***

  - Mettre en œuvre la **méthodologie en cascade**, de l'analyse des besoins jusqu'à la soutenance finale.
  - Développer les **compétences de travail en équipe**, en répartissant clairement les tâches et les responsabilités.
  - S'initier à la **rédaction de documents techniques** et à la planification d'un projet complet.

- ***Objectifs techniques***

  Concevoir une **plateforme web collaborative** permettant à plusieurs utilisateurs de :

  - Regarder une même vidéo de façon synchronisée ;
  - Échanger via un **chat temps réel** intégré ;
  - Créer et gérer des **salons virtuels** personnalisés.
  - Livrer un **MVP (Minimum Viable Product)** stable et fonctionnel à la fin du semestre.
  - Préparer une base solide pour le développement de futures fonctionnalités :
    - FAQ, multilingue, système d'abonnement, amélioration de la sécurité, etc.

1.4 Portée et limites du cahier des charges

Ce cahier des charges couvre les éléments suivants :

- La **présentation du projet** et de ses principales fonctionnalités ;
- Les **choix techniques** retenus pour le développement du MVP ;
- L'**analyse de faisabilité** et l'identification des risques ;
- L'**organisation du travail** et la **planification prévisionnelle**.

Cependant, le projet **n'inclut pas** :

- Le déploiement à grande échelle ou la mise en production définitive ;
- Le développement d'une **application mobile native** ;
- Les fonctionnalités avancées non essentielles à la première version (paiements, comptes premium, etc.).

Ces aspects pourront être envisagés lors d'évolutions ultérieures du projet.

1.5 Définitions et abréviations

Terme / Abréviation	Définition
Salon	Espace virtuel où plusieurs utilisateurs regardent une vidéo ensemble et interagissent.
Chat	Messagerie instantanée intégrée au salon.
Playlist	Liste de vidéos collaborative ajoutée par les membres du salon.
MVP	Version minimale fonctionnelle contenant les fonctionnalités essentielles.
API	Interface permettant à l'application de communiquer avec des services externes (ex : YouTube).
Front-end	Partie visible de l'application développée avec React.js et Tailwind CSS.
Back-end	Partie serveur développée avec Laravel, gérant la logique métier et les données.
Client	Application côté navigateur (React) utilisée par l'utilisateur.
Serveur	Partie du système qui traite les requêtes et gère les données.
SQL	Langage de gestion des bases de données relationnelles.
MySQL	Base de données utilisée pour stocker utilisateurs, salons et messages.
React.js	Bibliothèque JavaScript utilisée pour construire l'interface utilisateur.
Tailwind CSS	Framework CSS facilitant la création d'un design moderne et responsive.
Laravel	Framework PHP structuré (MVC) pour gérer les routes API, la sécurité et les données.
JWT	Jeton sécurisé utilisé pour l'authentification des utilisateurs.
UML	Langage de modélisation décrivant le fonctionnement du système.
Diagramme de cas d'utilisation	Diagramme UML montrant les interactions entre acteurs et système.
Diagramme de classes	Diagramme UML représentant les entités du système et leurs relations.
Diagramme de Gantt	Planning visuel des tâches du projet.
VIT	Exigence vitale, indispensable pour le MVP.
IMP	Exigence importante mais non critique.
MIN	Exigence mineure prévue pour une évolution future.

2. Contenu du projet

2.1 Fonctionnalités (MVP)

Fonctionnalité	Description reformulée	Priorité
Inscription et authentification	Création de compte, connexion sécurisée (email + mot de passe), confirmation par e-mail, gestion du profil utilisateur.	VIT
Consultation des salons publics	Permettre aux visiteurs de voir les salons accessibles avant connexion (nom, nombre d'utilisateurs, informations générales).	VIT
Création de salons	Un utilisateur connecté peut créer un salon avec un nom, une description et un mot de passe facultatif.	VIT
Gestion des salons	Administration complète d'un salon : permissions, contrôle vidéo, gestion des sondages, gestion des membres, gestion des vidéos.	VIT
Rejoindre un salon	Entrée dans un salon via un lien ou un code d'accès.	VIT
Acceptation des conditions du salon	L'utilisateur doit accepter les règles du salon avant de participer.	VIT
Quitter un salon	Déconnexion du salon et retrait de la liste des participants.	VIT
Chat en temps réel	Messagerie intégrée permettant l'envoi et la réception immédiate de messages.	VIT
Réactions & interactions sociales	Réactions emoji, notation du salon, participation aux sondages.	IMP
Visionnage synchronisé des vidéos YouTube	Lecture/pause/avance synchronisés pour tous les membres via l'API YouTube IFrame.	VIT
Synchronisation des actions vidéo	Transmission automatique des actions vidéo (lecture, pause, reprise, changement).	VIT
Gestion des vidéos	Ajouter une vidéo (lien YouTube), supprimer une vidéo, changer la vidéo en cours.	VIT
Gestion des membres	Voir la liste des membres, supprimer un membre, bloquer un membre.	VIT
Gestion des permissions	Autoriser ou interdire le chat, le contrôle vidéo invité, les réactions.	VIT
Création et gestion des sondages	Créer un sondage, voter, consulter les résultats.	IMP
Multilingue	Interface disponible en plusieurs langues (français / anglais).	IMP
Mode sombre	Thème sombre pour améliorer le confort visuel.	MIN
FAQ / Aide utilisateur	Section d'aide intégrée à la plateforme.	MIN
Notifications locales	Alerte discrète lors d'un nouveau message ou d'un nouvel utilisateur.	MIN

2.2 Description des cas d'utilisation

2.2.1 Use cases – Visiteur

➤ UC01 – Inscription d'un nouvel utilisateur

- **Objectif :** Permettre à un visiteur de créer un compte sécurisé.
- **Lien avec le système :** Création d'un nouvel utilisateur + envoi d'un e-mail de confirmation.
- **Acteur principal :** Visiteur
- **Pré-condition :** Le visiteur n'a pas encore de compte.
- **Scénario :**
  1. Le visiteur clique sur *S'inscrire*.
  2. Il saisit son nom, son e-mail et un mot de passe.
  3. Il valide le formulaire.
  4. Le système enregistre les informations.
  5. Le système envoie un e-mail de confirmation.
- **Post-condition :** Le compte est créé mais reste inactif tant que le mail n'est pas confirmé.

➤ UC02 – Confirmer l'inscription

- **Objectif :** Activer un compte nouvellement créé.
- **Lien avec le système :** Validation d'un lien d'activation envoyé par e-mail.
- **Acteur principal :** Visiteur
- **Pré-condition :** L'inscription a été réalisée.
- **Scénario :**
  1. Le visiteur ouvre l'e-mail reçu.
  2. Il clique sur le lien d'activation.
  3. Le système active le compte.
- **Post-condition :** Le compte devient utilisable.

➤ UC03 – Connexion à la plateforme

- **Objectif :** Permettre à un utilisateur existant de se connecter.
- **Lien avec le système :** Vérification e-mail/mot de passe + création de session.
- **Acteur principal :** Visiteur / Utilisateur inscrit
- **Pré-condition :** Le compte est actif.
- **Scénario :**
  1. L'utilisateur ouvre *Se connecter*.
  2. Il saisit son e-mail et son mot de passe.
  3. Le système vérifie les identifiants.
  4. Si incorrect, un message d'erreur apparaît.
  5. Si correct, la session est ouverte.
- **Post-condition :** L'utilisateur accède à l'interface.

➤ UC04 – Consulter les salons publics

• **Objectif :** Permettre à un visiteur de visualiser les salons publics avant de rejoindre.

• **Lien avec le système :** Lecture des informations des salons disponibles.

• **Acteur principal :** Visiteur

• **Pré-condition :** Aucun.

• **Scénario :**

1. Le visiteur clique sur *Salons publics*.
2. Le système affiche la liste des salons publics.
3. Le visiteur sélectionne un salon.
4. Le système affiche les informations générales.

• **Post-condition :** Le visiteur obtient une vue d'ensemble du salon choisi.

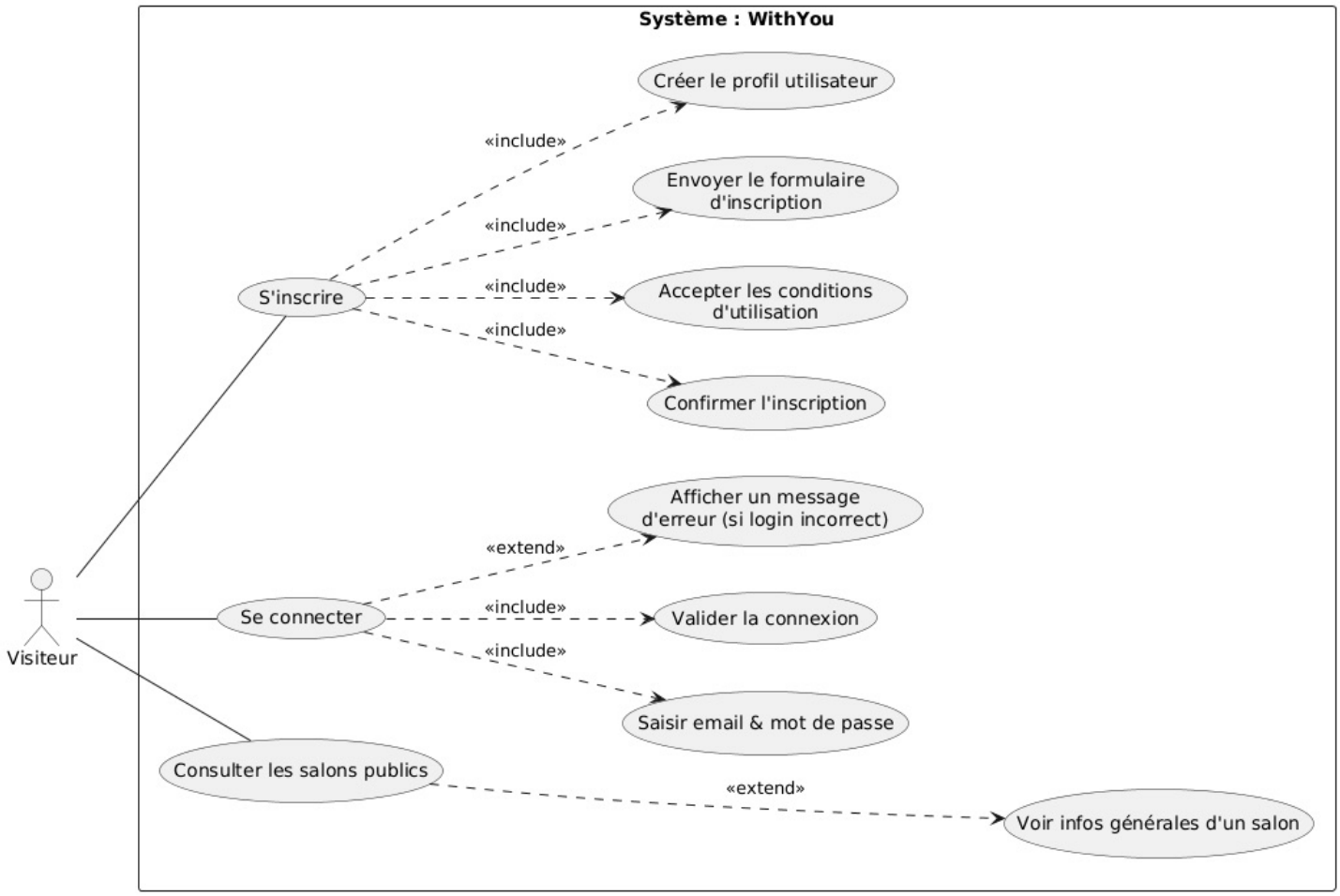

Figure 1 – Diagramme de cas d'utilisation : Visiteur

2.2.2 Use cases – Utilisateur Invité

➤ UC05 – Rejoindre un salon

• **Objectif :** Permettre à un utilisateur d'entrer dans un salon existant.

• **Lien avec le système :** Vérification du code ou du lien + ajout dans la liste des membres.

• **Acteur principal :** Utilisateur invité

• **Pré-condition :** L'utilisateur est connecté.

• **Scénario :**

1. L'utilisateur clique sur *Rejoindre un salon*.
2. Il saisit ou colle un code d'invitation.
3. Le système vérifie la validité du salon.
4. Le système vérifie les permissions du salon.
5. L'accès est accordé.

• **Post-condition :** L'utilisateur devient membre invité du salon.

➤ UC06 – Entrer dans le salon

• **Objectif :** Charger l'interface vidéo + chat du salon.

• **Lien avec le système :** Chargement de l'espace collaboratif (lecture vidéo + chat).

• **Acteur principal :** Utilisateur invité

• **Pré-condition :** L'accès au salon est autorisé.

• **Scénario :**

1. Le système charge l'interface vidéo.
2. Le système charge le chat du salon.

• **Post-condition :** L'utilisateur entre dans l'environnement partagé.

➤ UC07 – Synchronisation automatique de la vidéo

• **Objectif :** Reproduire la vidéo en synchronisation pour tous les membres.

• **Lien avec le système :** Mise à jour automatique via webhooks internes / signaux de synchro.

• **Acteur principal :** Utilisateur invité

• **Pré-condition :** L'utilisateur est dans le salon.

• **Scénario :**

1. La vidéo se lance automatiquement.
2. Le lecteur se synchronise en fonction de l'état global du salon.

• **Post-condition :** L'utilisateur voit la vidéo parfaitement synchronisée.

➤ UC08 – Interagir avec la vidéo

• **Objectif :** Permettre à l'utilisateur d'interagir localement avec la vidéo.

• **Lien avec le système :** Mise à jour locale ou globale selon permissions.

• **Acteur principal :** Utilisateur invité

• **Pré-condition :** L'utilisateur est dans le salon.

• **Scénario :**

1. L'utilisateur ajuste le volume.
2. L'utilisateur change la qualité.
3. L'utilisateur reprend la vidéo.
4. L'utilisateur met en pause (si autorisé).

• **Post-condition :** Les actions sont appliquées localement ou globalement selon le rôle.

➤ UC09 – Chatter et réagir

• **Objectif :** Envoyer des messages ou réactions emoji en temps réel.

• **Lien avec le système :** Envoi + diffusion des messages auprès des membres.

• **Acteur principal :** Utilisateur invité

• **Pré-condition :** L'utilisateur est dans le salon.

• **Scénario :**

1. L'utilisateur écrit un message.
2. Il envoie le message.
3. Les autres membres le reçoivent.

• **Post-condition :** Le message apparaît dans le chat.

➤ UC10 – Participer aux sondages et à la notation

• **Objectif :** Interagir avec les sondages du salon.

• **Lien avec le système :** Enregistrement de la réponse / note.

• **Acteur principal :** Utilisateur invité

• **Pré-condition :** Un sondage est disponible dans le salon.

• **Scénario :**

1. L'utilisateur répond au sondage.
2. L'utilisateur attribue éventuellement une note.
3. Le système enregistre la participation.

• **Post-condition :** La réponse est sauvegardée.

➤ UC11 – Quitter le salon

• **Objectif :** Quitter l'espace de visionnage.

• **Lien avec le système :** Mise à jour de la liste des membres.

• **Acteur principal :** Utilisateur invité

• **Pré-condition :** L'utilisateur est dans un salon.

• **Scénario :**

1. L'utilisateur clique sur *Quitter le salon*.
2. Le système le retire de la liste des membres.

• **Post-condition :** L'utilisateur n'a plus accès au salon.

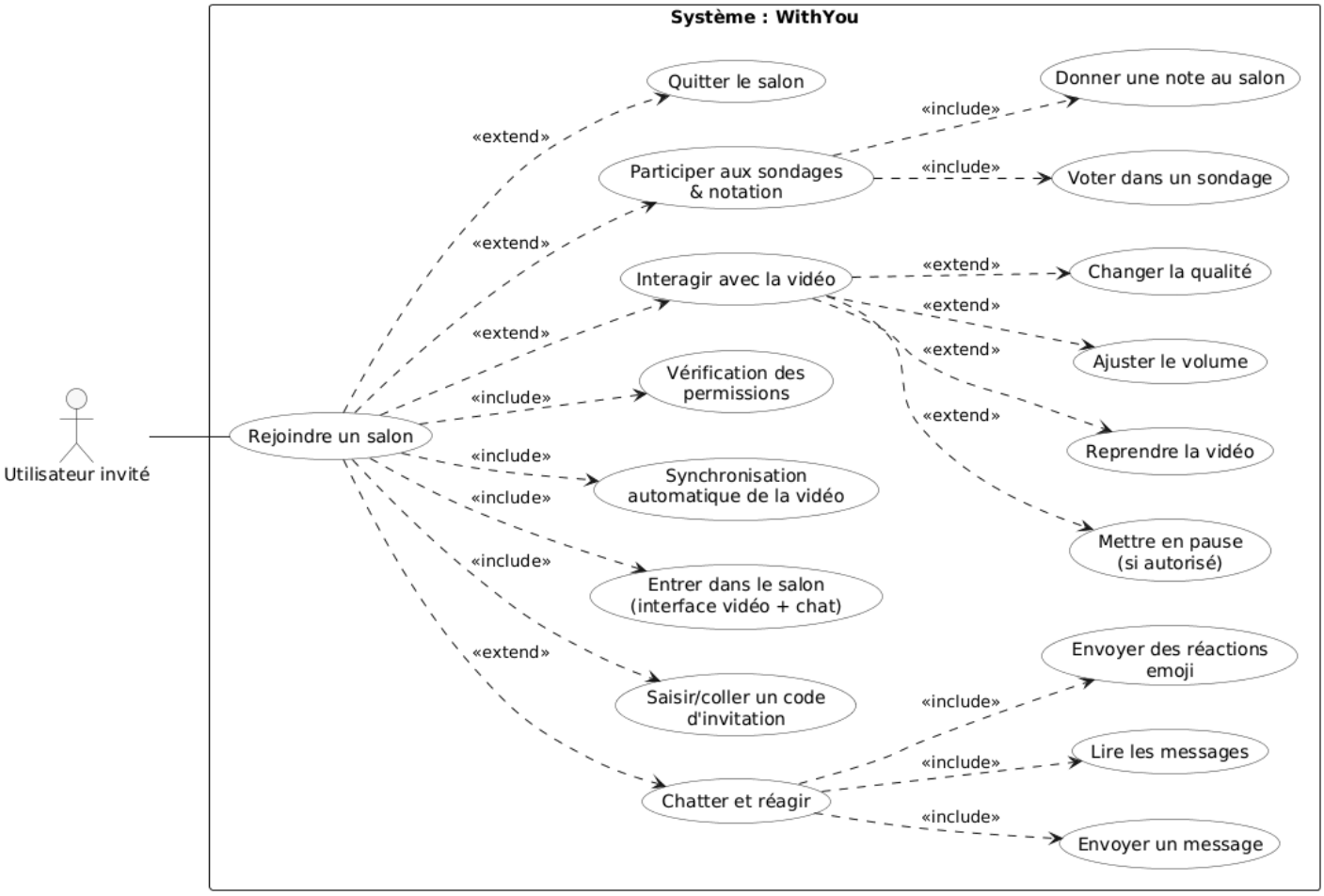

Figure 2 – Diagramme de cas d'utilisation : Utilisateur Invité

2.2.2 Use cases – Administrateur du salon

➤ UC12 – Créer un salon

• **Objectif :** Créer un nouvel espace de visionnage.

• **Lien avec le système :** Enregistrement d'un nouveau salon.

• **Acteur principal :** Administrateur

• **Pré-condition :** L'utilisateur est connecté.

• **Scénario :**

1. L'utilisateur clique *Créer un salon*.
2. Le système génère un espace vierge.

• **Post-condition :** Le salon est créé.

➤ UC13 – Entrer les informations du salon

• **Objectif :** Définir les paramètres du salon.

• **Lien avec le système :** Enregistrement du nom, description, vidéo initiale.

• **Acteur principal :** Administrateur

• **Pré-condition :** Salon créé.

• **Scénario :**

1. L'administrateur saisit le nom.
2. Il ajoute une description.
3. Il configure mot de passe / vidéo initiale.
4. Le système enregistre.

• **Post-condition :** Le salon est prêt à l'utilisation.

➤ UC14 – Gérer les vidéos

• **Objectif :** Modifier la playlist du salon.

• **Lien avec le système :** Mise à jour de la playlist.

• **Acteur principal :** Administrateur

• **Pré-condition :** Le salon contient une vidéo.

• **Scénario :**

1. Ajouter une vidéo (lien YouTube).
2. Supprimer une vidéo existante.
3. Changer la vidéo en cours.

• **Post-condition :** La playlist est mise à jour.

➤ UC15 – Gérer les permissions

• **Objectif :** Contrôler ce que les invités peuvent faire.

• **Lien avec le système :** Mise à jour des droits.

• **Acteur principal :** Administrateur

• **Pré-condition :** Le salon existe.

• **Scénario :**

1. L'administrateur ouvre *Gestion des permissions*.
2. Il active/désactive le chat.
3. Il autorise ou non le contrôle vidéo invité.
4. Il gère les codes d'invitation.

• **Post-condition :** Les permissions sont appliquées.

➤ UC16 – Gérer les membres

• **Objectif :** Contrôler les utilisateurs présents.

• **Lien avec le système :** Mise à jour de la liste des membres.

• **Acteur principal :** Administrateur

• **Pré-condition :** Des membres sont présents.

• **Scénario :**

1. Voir la liste des membres.
2. Supprimer un membre.
3. Bloquer un membre.

• **Post-condition :** La liste est actualisée.

➤ UC17 – Gérer les sondages

• **Objectif :** Créer et gérer les sondages du salon.

• **Lien avec le système :** Enregistrement des sondages + consultation des résultats.

• **Acteur principal :** Administrateur

• **Pré-condition :** Administrateur connecté.

• **Scénario :**

1. Créer un sondage.
2. Le système enregistre.
3. Consulter les résultats.

• **Post-condition :** Les sondages sont disponibles.

➤ UC18 – Contrôler la vidéo globalement

• **Objectif :** Gérer la lecture vidéo pour tous les membres.

• **Lien avec le système :** Mise à jour globale de l'état de la vidéo.

• **Acteur principal :** Administrateur

• **Pré-condition :** Le salon existe.

• **Scénario :**

1. Mettre la vidéo en pause pour tous.
2. Reprendre la vidéo pour tous.
3. Synchroniser la vidéo.

• **Post-condition :** La vidéo est synchronisée pour tous les membres.

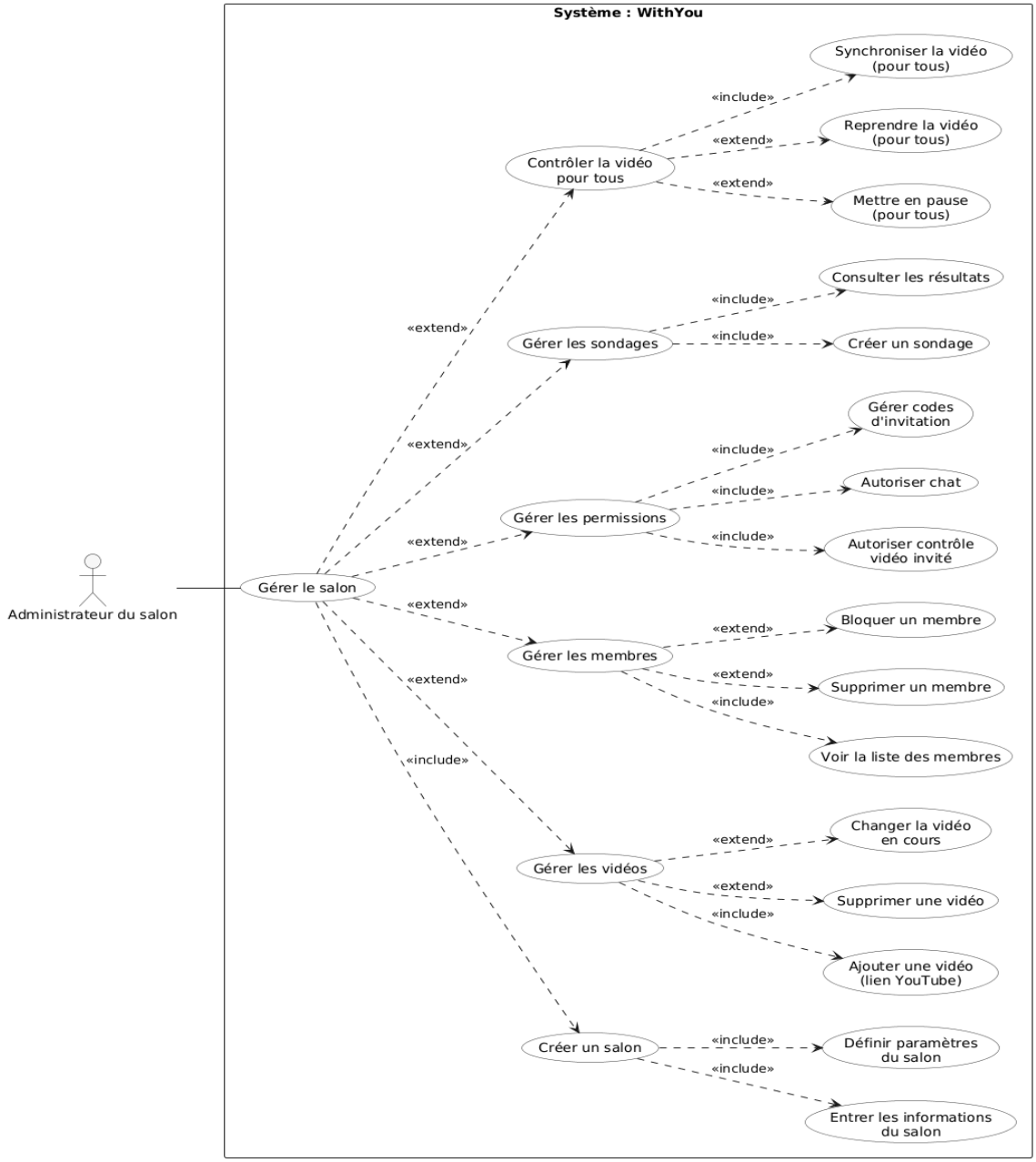

Figure 3 – Diagramme de cas d'utilisation : Administrateur du salon

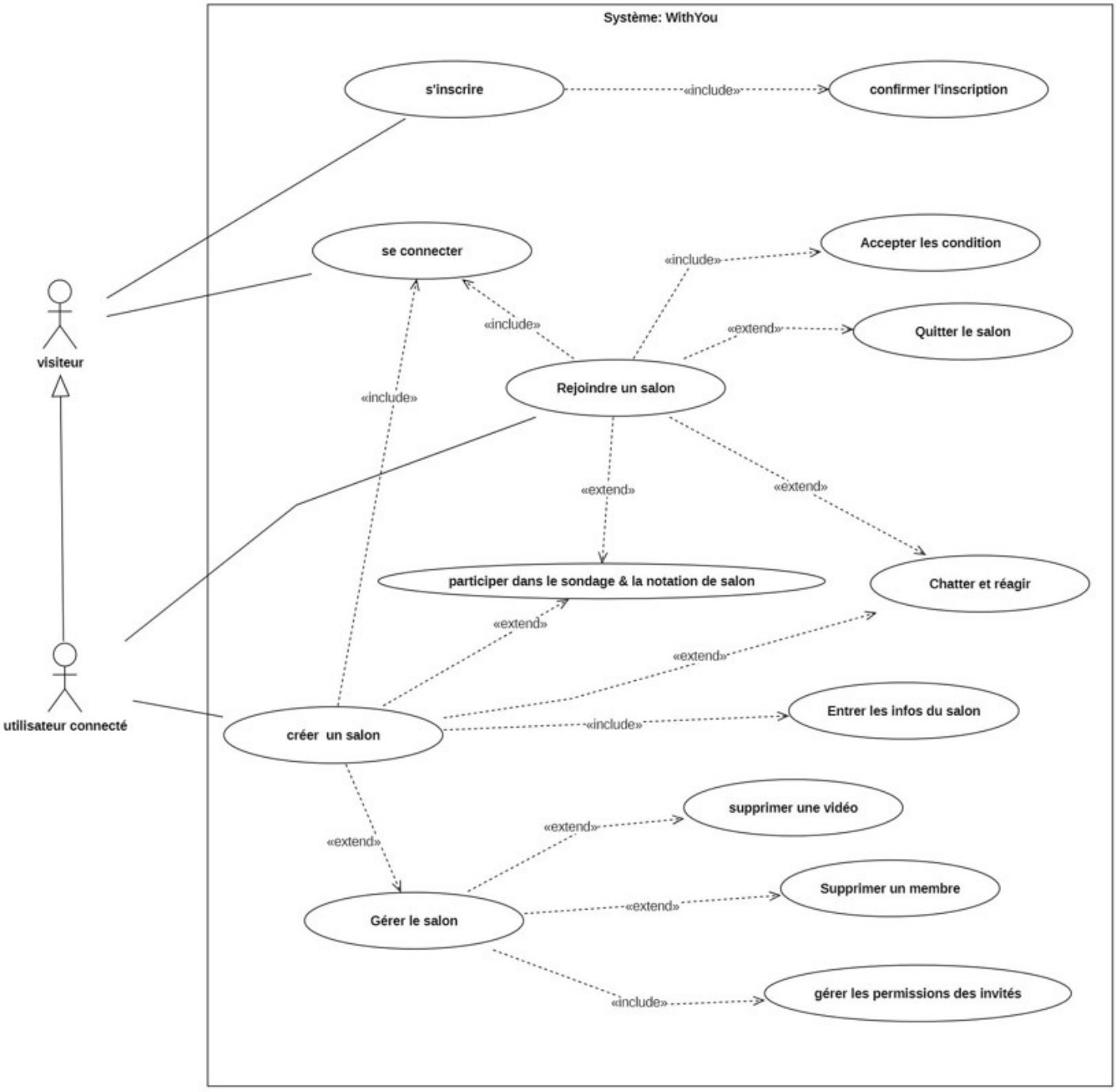

Figure 4 – Diagramme de cas d'utilisation global du système

2.3 Choix techniques et justification des technologies

Le choix des technologies utilisées pour le développement de *With U* repose sur des critères de cohérence technique, de simplicité d'intégration et de maîtrise par l'équipe.

Chaque outil a été sélectionné afin d'assurer un équilibre entre **performance**, **maintenabilité** et **facilité de collaboration** au sein du groupe.

L'architecture technique adoptée suit un modèle **client–serveur**, séparant clairement :

- la partie **front-end** (interface utilisateur),
- la partie **back-end** (traitement des données, logique métier).

• Front-end : React.js & Tailwind CSS

L'interface utilisateur est développée avec **React.js**, uReact et l'API Laravele permettant de créer des composants dynamiques et réutilisables.

Le style est assuré par **Tailwind CSS**, un framework léger offrant un design propre, responsive et simple à maintenir.

• Back-end : PHP (Laravel Framework)

La partie serveur repose sur **Laravel**, un framework PHP fiable et bien structuré (modèle MVC).

Il facilite la création d'une **API REST** et gère nativement les routes, l'authentification, la sécurité et la connexion à la base de données.

Laravel offre un bon équilibre entre simplicité, robustesse et compatibilité avec le front-end en React.

• Base de données : MySQL

Les données (utilisateurs, salons, messages, vidéos) sont stockées dans **MySQL**, un système relationnel stable et compatible avec Laravel.

L'ORM **Eloquent** facilite la gestion des requêtes, assure la cohérence des données et renforce la fiabilité du stockage.

• API externe : YouTube IFrame API

L'API IFrame de YouTube permet d'intégrer et contrôler les vidéos directement depuis YouTube, garantissant une lecture fluide sans gestion de stockage vidéo.

• Sécurité : JSON Web Token (JWT)

L'authentification repose sur les **JWT**, qui génèrent un jeton sécurisé pour chaque utilisateur connecté.

Cette méthode est légère, fiable et parfaitement adaptée aux échanges entre le front-end en React et l'API Laravel.

- **Environnement de développement et outils collaboratifs**

L'environnement de travail s'appuie sur un ensemble d'outils assurant la collaboration, la planification et la version du code :

- **GitHub** : hébergement du code source, gestion des branches et suivi des versions.
- **Jira Software** : gestion des tâches, planification des sprints et suivi de l'avancement du projet.
- **Notion / Google Docs** : centralisation de la documentation et des réunions d'équipe.
- **Postman** : test et validation des routes API pendant le développement.
- **Figma** : création de maquettes et de diagrammes UML.

Cette organisation garantit une **traçabilité complète du projet**, une **collaboration fluide** entre les membres, et une **meilleure visibilité de l'avancement** des différentes phases.

Synthèse du choix technologique

Le système d'authentification repose sur l'utilisation de JSON Web Token (JWT).

Chaque utilisateur possède un jeton unique généré par le serveur Laravel après connexion, lui permettant d'accéder à son compte et à ses données de manière sécurisée.

Cette méthode est légère, fiable et compatible avec les API REST, facilitant la communication entre le front React et le back PHP.

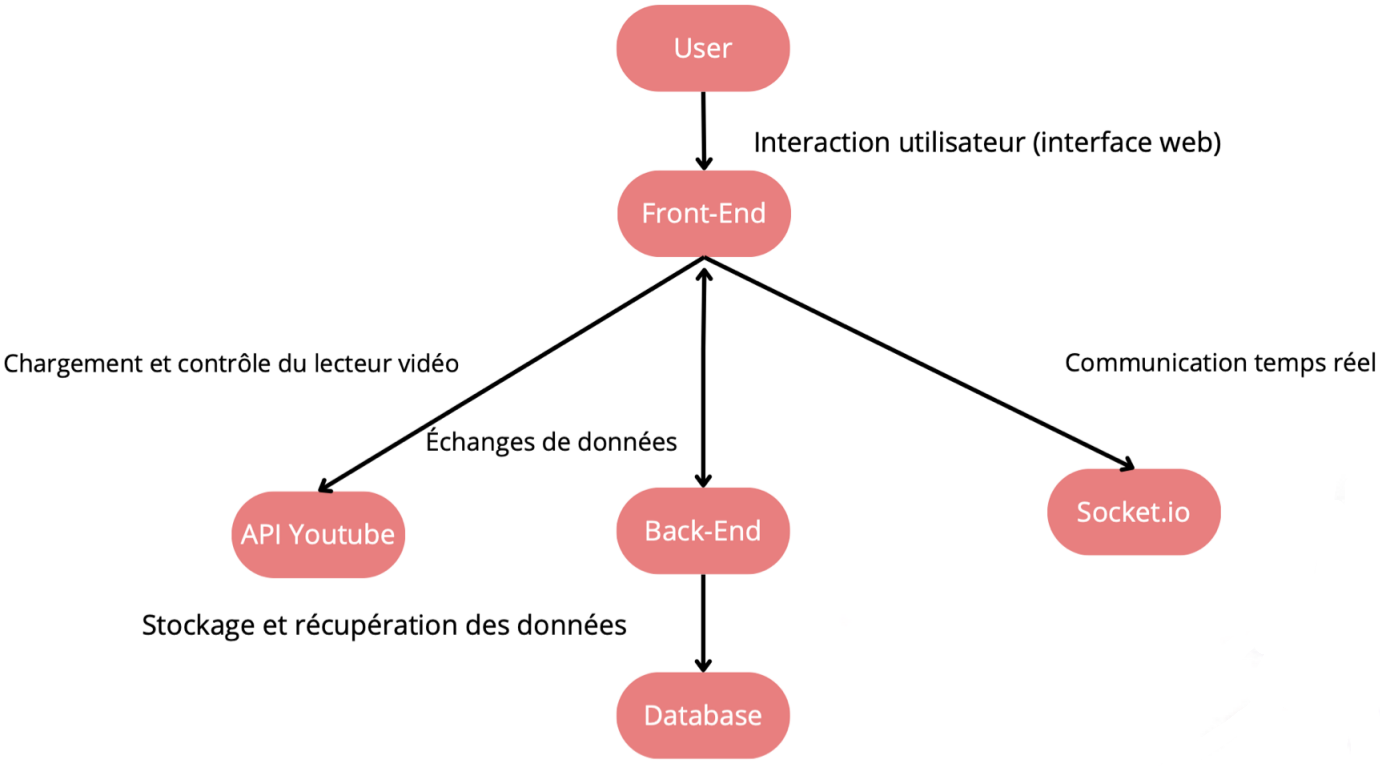

Figure 5 – Schéma d'architecture technique

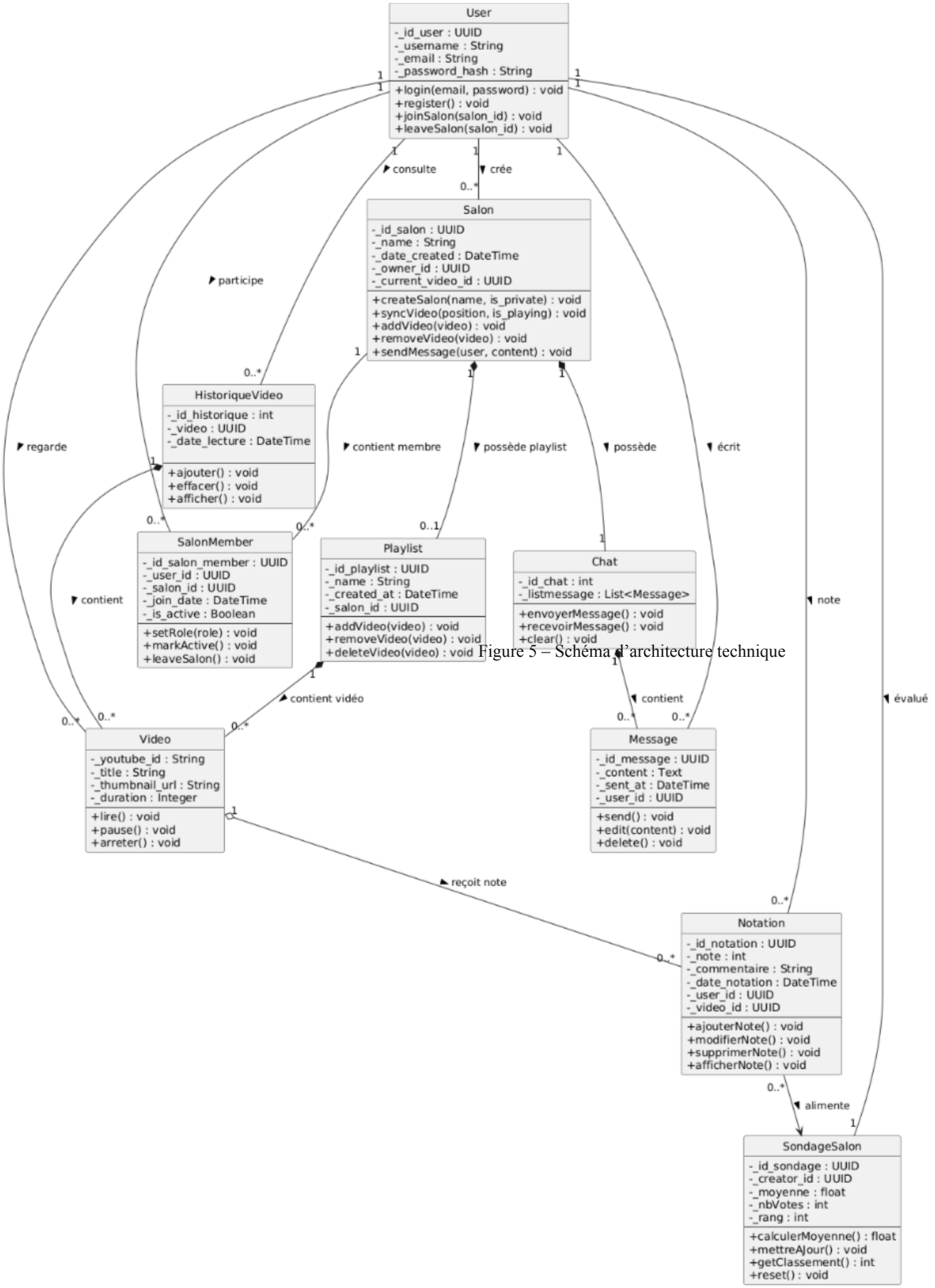

Figure 5 – Schéma d'architecture technique

Figure 6 – Modélisation conceptuelle - Diagramme de classes UML

2.4 Contraintes et risques identifiés

Malgré la faisabilité confirmée, certaines contraintes techniques et organisationnelles peuvent influencer le déroulement du projet.

1.Contraintes techniques

Dépendance à l'API YouTube : les quotas de requêtes et la stabilité du service peuvent limiter les tests intensifs.
→ Une solution de repli sera prévue en cas d'indisponibilité.

Gestion du temps réel : la latence réseau peut entraîner des décalages de synchronisation entre les utilisateurs.
→ Des tests précoces et itératifs permettront d'ajuster la logique Socket.io.

2. Contraintes temporelles

Le projet doit être achevé en 16 semaines, ce qui impose une hiérarchisation stricte des priorités.
→ Les fonctionnalités secondaires seront développées uniquement si le MVP est terminé à temps.

3. Contraintes humaines

Les membres de l'équipe travaille sur différents systèmes (Windows, macOS).
→ Un environnement commun standardisé a été mis en place pour éviter les incompatibilités.

La coordination à distance nécessite une communication rigoureuse.
→ Des réunions hebdomadaires et des bilans sur Jira assurent le suivi collectif.

2.5 Mesures de prévention et gestion des risques

L'équipe a identifié les principaux risques susceptibles d'affecter le projet et mis en place des **stratégies de prévention adaptées** :

Type de risque	Description	Mesures préventives
Technique	Échec ou blocage de l'API YouTube	Limiter les appels à l'API
Synchronisation	Désynchronisation des vidéos entre utilisateurs	Tests réguliers de Socket.io et ajustement du délai réseau
Organisationnel	Retard dans certaines fonctionnalités	Suivi hebdomadaire sur Jira et réévaluation du planning
Communication	Manque de coordination entre membres	Réunions hebdomadaires et échanges constants sur WhatsApp
Versioning	Conflits GitHub ou perte de code	Utilisation de branches dédiées et validation systématique des pull requests

Grâce à ces mesures, les risques identifiés sont **anticipés et maîtrisés**.

3. Organisation et gestion du projet

L'organisation du projet With U s'appuie sur une méthodologie rigoureuse, une répartition claire des responsabilités et une planification structurée.
Elle a été conJirapour assurer un déroulement fluide, une communication constante entre les membres et une livraison du produit final dans les délais impartis.
L'équipe s'est inspirée du modèle en cascade, adapté aux contraintes académiques, tout en intégrant certaines pratiques issues de la gestion agile (sprints, suivi Jira, réunions hebdomadaires).

3.1 Méthodologie de travail (modèle en cascade)

Le projet suit une **méthode en cascade**, adaptée au cadre universitaire.

Chaque étape est réalisée dans un ordre logique et doit être validée avant de passer à la suivante, garantissant une progression claire et maîtrisée.

Pour faciliter la coordination, l'équipe applique quelques pratiques agiles :

- **Sprints hebdomadaires**,
- **Bilans réguliers**,
- **Outils collaboratifs** (Jira, GitHub, Notion, Excel).

Les grandes étapes du projet sont :

- **Analyse et cadrage**
- **Conception et modélisation**
- **Mise en place technique**
- **Développement du MVP**
- **Tests et validation**
- **Soutenance et documentation finale**

3.2 Répartition des rôles et responsabilités

L'organisation interne du groupe a été définie de manière à exploiter au mieux les compétences techniques de chaque membre. Chaque étudiant occupe un rôle précis et contribue à la réalisation du projet selon son domaine d'expertise.

Malek Ghabi — Coordination & Front-end

1 - Planifier les sprints, organiser le travail sur Jira et suivre l'avancement.

2 - Rédiger le cahier des charges et structurer la documentation sur Notion.

3 - Concevoir les maquettes graphiques (interface, salon, pages principales..).

4 - Développer l'interface utilisateur en React.js (pages, navigation..).

5 - Intégrer le lecteur YouTube et assurer la cohérence visuelle globale.

6 - Effectuer les tests UI et ajustements graphiques avant intégration finale.

Meriem Takdjerad — Front-end logique & Intégration API

1 - Concevoir l'architecture front-end et les interactions de l'application.

2 - Développer la logique des formulaires (inscription, connexion).

3- Intégrer les API du back-end pour les salons, les messages et les utilisateurs.

4 - Gérer l'authentification front (gestion du token, redirections).

5 - Développer l'affichage des données : salons, messages, utilisateurs.

6 - Réaliser les tests d'intégration front/back et corriger les anomalies.

Wissam Taleb — Back-end Laravel & Base de données

1 - Initialiser et configurer le back-end Laravel.

2 - Concevoir la base de données et créer les tables nécessaires.

3 - Développer les routes API (salons, utilisateurs, messages).

4 - Implémenter l'authentification sécurisée avec JWT.

5 - Garantir la cohérence et la fiabilité des données stockées.

6 - Tester et valider les routes via Postman, puis corriger les bugs.

Yanis Laftimi — Back-end & Logique métier

1 - Développer la logique métier Laravel (gestion salons, permissions).

2 - Gérer les rôles utilisateurs : créateur, invité, accès aux ressources.

3 - Optimiser les performances du serveur (requêtes, contrôleurs).

4 - Participer au développement des API internes (création, mise à jour, suppression).

5 - Collaborer aux tests API avec Wissam pour garantir la stabilité.

6 - Assurer la correction des anomalies backend et la validation finale.

Figure 7– Planning du projet — Diagramme de Gantt

4. Conclusion

4.1 Synthèse du cahier des charges

Le présent cahier des charges réunit l'ensemble des éléments nécessaires à la compréhension et à la conception du projet **With U** : objectifs fonctionnels, description détaillée des cas d'utilisation, modélisation UML, architecture technique, critères de qualité et planification du développement.

Les choix technologiques retenus — **React.js** et **Tailwind CSS** pour le front-end, **Laravel (PHP)** pour le back-end, **MySQL** pour la base de données, ainsi que l'intégration de l'**API YouTube IFrame** — assurent une architecture moderne, cohérente et adaptée aux besoins du projet.

Ces technologies offrent un bon équilibre entre performance, maintenabilité et accessibilité, permettant à l'équipe de concevoir une application fiable et évolutive dans les délais impartis.

4.2 Perspectives et évolutions futures

Plusieurs pistes d'amélioration pourront être envisagées dans les versions futures de With U :

➢ **Fonctionnel**

- Ajout de profils utilisateurs enrichis (photo, biographie, préférences).
- Mise en place de rôles avancés dans les salons (modérateur, co-administrateur).
- Historique des vidéos visionnées et recommandations personnalisées.
- Amélioration du système de sondages et d'interactions sociales.

➢ **Qualité**

- Mise en place de tests unitaires, d'intégration et end-to-end.
- Ajout de logs détaillés et d'un système de monitoring.
- Amélioration de l'expérience utilisateur via des tests ergonomiques.

➢ **Performance**

- Optimisation des requêtes SQL et du chargement vidéo.
- Mise en cache de certaines données (Redis).
- Optimisation de la synchronisation et de la gestion des salons.

➢ **Sécurité**

- Mise en place d'un système de refresh token (JWT).
- Renforcement des règles de sécurité (CORS, rate limiting, CSP).
- Amélioration du système d'invitations et de permissions.

➢ **Déploiement**

- Automatisation du pipeline CI/CD via GitHub Actions.
- Hébergement sur une plateforme managée (Render, Railway, Hostinger…).
- Possibilité d'utiliser un **nom de domaine personnalisé** pour une meilleure identité visuelle.


With U