CAHIER DES CHARGES

Projet IL 2025/2026

Plateforme collaborative de visionnage vidéo

Prestataires

Taleb Wissame - Takdjerad Meriem - Ghabi Malek - Laftimi Yanis

Clients

Rémy Kesller - Ludovic Bonnefoy

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 peuvent se réunir pour regarder une vidéo en simultané et échanger via un chat intégré.
Chat	Système de messagerie instantanée intégré à la plateforme, permettant la communication entre les utilisateurs d'un salon.
Playlist	Liste de vidéos collaborative pouvant être créée et modifiée par les membres d'un salon.
MVP (Minimum Viable Product)	Version minimale fonctionnelle du produit contenant les fonctionnalités essentielles à sa validation.
API (Application Programming Interface)	Interface logicielle permettant à l'application de communiquer avec des services externes, comme l'API YouTube.
Front-end	Partie visible de l'application, correspondant à l'interface utilisateur développée avec des technologies web (HTML, CSS, React.js).
Back-end	Partie invisible du système, chargée de gérer la logique serveur, la base de données et la communication avec le front-end.
Client	Application exécutée dans le navigateur de l'utilisateur (front-end), communiquant avec le serveur.
Serveur	Partie du système qui exécute les requêtes des clients et gère les échanges de données.
SQL	Langage de requête structuré permettant la création, la manipulation et la gestion de bases de données relationnelles.
PostgreSQL	Système de gestion de base de données relationnelle open-source utilisé pour stocker les informations des utilisateurs et des salons.
React.js	Bibliothèque JavaScript moderne utilisée pour le développement de l'interface utilisateur et la gestion des composants dynamiques.
Node.js / Express	Environnement serveur et framework JavaScript permettant de développer des applications web rapides et légères côté back-end.
Socket.io	Bibliothèque facilitant la communication en temps réel entre le serveur et les clients (utilisée pour la synchronisation vidéo et le chat).
JWT (JSON Web Token)	Méthode d'authentification sécurisée permettant de vérifier l'identité d'un utilisateur via un jeton unique.
UML (Unified Modeling Language)	Langage de modélisation utilisé pour représenter graphiquement les fonctionnalités et la structure du système (diagrammes de cas d'utilisation, classes, séquence).
Diagramme de Gantt	Outil de planification représentant visuellement les tâches du projet et leur durée dans le temps.
VIT	Exigence vitale : indispensable au bon fonctionnement du MVP.
IMP	Exigence importante : souhaitée mais non critique pour la première version.
MIN	Exigence mineure : prévue pour une évolution future du projet.

2. Contenu du projet

2.1 Présentation générale de la plateforme

Le projet **With U** consiste en la conception et le développement d'une plateforme web collaborative permettant à plusieurs utilisateurs de regarder des vidéos en ligne **de manière synchronisée**, tout en **interagissant via un chat intégré**.

L'objectif principal est de **recréer une expérience collective de visionnage**, accessible depuis n'importe quel appareil connecté à Internet.

Chaque utilisateur pourra **créer un salon virtuel**, y inviter d'autres personnes et interagir en temps réel autour du contenu vidéo.

Cette plateforme repose sur l'idée que le partage audiovisuel, au-delà du simple visionnage, peut devenir une **expérience sociale interactive**, combinant communication, synchronisation et convivialité.

Le projet s'inscrit dans un cadre académique et constitue une mise en application des connaissances acquises dans les domaines du **développement web full-stack**, de la **gestion de bases de données**, du **travail collaboratif** et de la **modélisation UML**.

Le développement de **With U** poursuit trois objectifs majeurs :

- **Offrir une expérience interactive et fluide** grâce à la synchronisation en temps réel des vidéos entre plusieurs utilisateurs.
- **Favoriser la communication instantanée** au sein d'un salon par un système de messagerie intégré et ergonomique.
- **Proposer une solution web moderne, accessible et sécurisée**, qui respecte les standards du développement web et les contraintes techniques du projet universitaire.

D'un point de vue pédagogique, ce projet permet également de **mettre en pratique l'ensemble des compétences** acquises en programmation, conception UML, administration système, travail d'équipe et gestion de projet agile.

2.2 Fonctionnalités principales (MVP)

Les fonctionnalités suivantes constituent le **noyau du projet**.

Elles sont indispensables à la première version livrable (MVP) et garantissent le bon fonctionnement de la plateforme.

Fonctionnalité	Description	Priorité
Inscription et authentification	Création de compte, connexion sécurisée (e-mail / mot de passe), déconnexion et récupération de mot de passe via lien de réinitialisation.	VIT
Création et gestion des salons	Création de salons virtuels avec un nom et un lien d'invitation. L'utilisateur peut rejoindre un salon existant, ou en supprimer un s'il en est le créateur.	VIT
Visionnage collectif de vidéos YouTube	Intégration du lecteur YouTube via l'API officielle. Lecture, pause, avance ou reprise synchronisées pour tous les membres d'un salon.	VIT
Chat en temps réel	Messagerie instantanée intégrée à chaque salon, permettant aux utilisateurs d'échanger pendant le visionnage.	VIT
Synchronisation entre utilisateurs	Transmission automatique des actions vidéo grâce à Socket.io (lecture, pause, reprise, changement de vidéo).	VIT

—> Ces fonctionnalités seront **implémentées et testées** en priorité afin d'obtenir un produit stable, conforme aux attentes du projet IL.

2.3 Fonctionnalités secondaires (évolutives)

Ces fonctionnalités ne sont pas indispensables au MVP, mais seront envisagées **si le temps et les ressources le permettent**, ou intégrées dans une version ultérieure.

Fonctionnalité	Description	Priorité
Multilingue	Possibilité de changer la langue de l'interface (français / anglais).	IMP
Système d'abonnement	Création d'un espace "Premium" avec options avancées (playlist personnelle, historique, thèmes).	MIN
FAQ / Aide utilisateur	Rubrique d'aide et de documentation intégrée pour répondre aux questions fréquentes.	IMP
Profil utilisateur	Modification du pseudonyme, photo de profil et paramètres du compte.	IMP
Mode sombre	Option de personnalisation visuelle pour améliorer le confort d'utilisation.	MIN

2.4 Description des cas d'utilisation (Use Cases)

2.4.1 Vue d'ensemble

Les cas d'utilisation décrivent toutes les interactions possibles entre les **acteurs** et le système *With U*.

Ils constituent la base fonctionnelle du projet et guideront la conception UML, le développement et les tests.

Acteurs identifiés :

- **Visiteur :** utilisateur non connecté qui peut consulter la page d'accueil et s'inscrire.
- **Utilisateur authentifié :** membre connecté qui accède aux salons, au chat et aux vidéos.
- **Créateur de salon :** utilisateur ayant créé un salon ; il dispose de droits spécifiques.
- **Administrateur :** utilisateur ayant des privilèges de modération (suppression, gestion des utilisateurs).

2.4.2 Liste complète des cas d'utilisation

	Cas d'utilisation	Description synthétique	Acteur principal
UC01	Créer un compte	Saisir ses informations et valider par e-mail pour accéder à la plateforme.	Visiteur
UC02	Se connecter	Authentifier un utilisateur existant (e-mail + mot de passe).	Visiteur
UC03	Réinitialiser son mot de passe	Envoyer un lien de réinitialisation par e-mail.	Visiteur
UC04	Gérer son profil	Modifier pseudo, mot de passe, langue ou photo de profil.	Utilisateur
UC05	Créer un salon	Créer un nouvel espace de visionnage avec nom, mot de passe et URL vidéo.	Utilisateur
UC06	Rejoindre un salon	Rejoindre un salon existant via un lien d'invitation ou un code.	Utilisateur
UC07	Visionnage synchronisé	Regarder une vidéo YouTube avec lecture/pause synchronisées pour tous.	Créateur / Utilisateur
UC08	Contrôler la lecture vidéo	Mettre en pause, reprendre, avancer ou reculer une vidéo.	Créateur
UC09	Ajouter une vidéo à la playlist	Ajouter ou supprimer des vidéos dans la file de lecture du salon.	Créateur / Utilisateur
UC10	Consulter la playlist	Voir les vidéos disponibles dans le salon.	Utilisateur
UC11	Discuter via le chat	Envoyer des messages instantanés et réagir avec des émojis.	Utilisateur
UC12	Recevoir des notifications	Être alerté lorsqu'un message est reçu ou qu'une nouvelle vidéo démarre.	Utilisateur
UC13	Changer la langue de l'interface	Sélectionner la langue d'affichage (français, anglais, arabe...).	Utilisateur
UC14	Consulter la FAQ	Lire la rubrique d'aide intégrée pour résoudre un problème courant.	Utilisateur
UC15	S'abonner à une offre	Choisir un abonnement (mensuel, annuel, premium) et activer ses avantages.	Utilisateur
UC16	Gérer les abonnements	Mettre à jour, suspendre ou résilier une formule d'abonnement.	Utilisateur
UC17	Supprimer ou quitter un salon	Quitter un salon ou le supprimer s'il en est le créateur.	Utilisateur / Créateur
UC18	Modérer un salon	Supprimer des messages inappropriés ou bannir un utilisateur.	Administrateur
UC19	Se déconnecter	Mettre fin à la session utilisateur de façon sécurisée.	Utilisateur

2.4.3 Description synthétique des principaux cas

UC01 – Créer un compte

Le visiteur saisit ses informations (nom, e-mail, mot de passe) et reçoit un e-mail de confirmation.

—>. **Objectif :** permettre l'accès sécurisé à la plateforme et éviter les comptes fictifs.

UC05 – Créer un salon

L'utilisateur connecté crée un salon, définit un nom et éventuellement un mot de passe, puis obtient un lien d'invitation à partager.

—>. **Objectif :** créer un espace virtuel de visionnage privé ou public.

UC07 – Visionnage synchronisé

Tous les participants regardent la même vidéo YouTube simultanément.

Les actions "play", "pause" ou "avance" du créateur sont répercutées automatiquement sur tous les écrans.

—> **Objectif :** offrir une expérience fluide et collective de visionnage.

UC11 – Chat en temps réel

Les membres du salon échangent des messages instantanément.

Les messages s'affichent en direct grâce à la technologie Socket.io.

—> **Objectif :** favoriser la communication et l'interactivité pendant le visionnage.

UC13 – Changer la langue de l'interface

L'utilisateur choisit la langue d'affichage dans les paramètres.

Le site se recharge automatiquement dans la langue sélectionnée.

—> **Objectif :** rendre la plateforme accessible à un public international.

UC15 – S'abonner à une offre

L'utilisateur choisit une formule d'abonnement (mensuelle, annuelle, premium) pour accéder à des fonctionnalités avancées (ex. : salons illimités, thèmes personnalisés).

—> **Objectif :** offrir une montée en gamme progressive et tester un modèle économique futur.

UC17 – Supprimer ou quitter un salon

Le créateur peut supprimer un salon à tout moment ; les autres utilisateurs peuvent le quitter librement.

—> **Objectif :** permettre la gestion et la fermeture propre des sessions.

UC18 – Modérer un salon

L'administrateur peut supprimer un message, expulser un membre ou fermer un salon problématique.

—> **Objectif :** garantir la sécurité et le respect au sein des salons publics.

2.4.4 Schéma UML associé

Le diagramme de cas d'utilisation, bien que prévu dans la phase de conception, **n'a pas été réalisé à ce stade du projet**.

L'équipe a choisi de **prioriser la rédaction du cahier des charges et la définition claire des fonctionnalités** afin d'assurer la cohérence du projet avant le passage à la modélisation UML.

Ce diagramme pourra être élaboré ultérieurement lors de la **phase de conception technique**, à partir des cas d'utilisation listés ci-dessus, afin de visualiser les interactions entre les acteurs et le système *With U*.

2.5 Choix techniques et justification des technologies

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

Chaque outil a été sélectionné de manière à garantir un équilibre entre **performance**, **maintenabilité** et **facilité de collaboration** au sein du groupe.

L'architecture technique adoptée suit le modèle **client–serveur**, avec une séparation claire entre la partie **front-end** (interface utilisateur) et la partie **back-end** (traitement des données et logique applicative).

Front-end : React.js & Tailwind CSS

L'interface utilisateur sera développée avec **React.js**, une bibliothèque JavaScript moderne, performante et simple à maintenir.

Elle permet de créer des composants réutilisables et dynamiques, essentiels pour une application interactive.

Le style sera assuré par **Tailwind CSS**, un framework léger favorisant un design propre et responsive.

Back-end : Node.js & Express.js

La partie serveur repose sur **Node.js** associé à **Express.js**.

Ce choix permet de créer une architecture rapide, modulaire et entièrement écrite en JavaScript, ce qui facilite la communication entre le front et le back.

Express simplifie la gestion des routes et des requêtes API, tandis que Node.js assure la performance et la stabilité du système.

Base de données : PostgreSQL

Les données (utilisateurs, salons, messages) seront stockées dans **PostgreSQL**, une base relationnelle fiable et sécurisée.

Elle garantit la cohérence et la stabilité des informations, tout en s'intégrant parfaitement avec Node.js.

Communication temps réel : Socket.io

Pour la synchronisation des vidéos et le chat instantané, la bibliothèque **Socket.io** a été retenue.

Elle permet une communication bidirectionnelle et immédiate entre le client et le serveur.

API externe : YouTube IFrame API

L'intégration de **l'API YouTube IFrame** permet de lire et contrôler des vidéos hébergées sur YouTube sans avoir à les stocker localement.

Cette solution garantit la légalité, la rapidité et la stabilité de la lecture vidéo.

Sécurité : JSON Web Token (JWT)

Le système d'authentification des utilisateurs utilise **JWT**, une méthode sécurisée et légère pour vérifier les connexions.

Chaque utilisateur possède un jeton unique lui permettant d'accéder à son compte sans risque d'usurpation.

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

L'ensemble des choix techniques repose sur une architecture web **moderne, performante et évolutive**.

Les technologies retenues (React, Node.js, PostgreSQL, Socket.io, JWT) sont toutes **open source**, bien documentées et compatibles entre elles.

Elles offrent une solution robuste répondant aux exigences du projet : **communication temps réel**, **sécurité**, **ergonomie**, et **simplicité de maintenance**.

2.6 Fiabilité, contraintes et risque

L'étude de faisabilité du projet **With U** a pour objectif d'évaluer la **viabilité technique**, **organisationnelle** et **temporelle**du développement de la plateforme.
Elle permet de s'assurer que le projet peut être réalisé dans les délais impartis, avec les ressources disponibles et les compétences de l'équipe.
Cette analyse identifie également les **principales contraintes** et les **risques potentiels** afin de mettre en place des mesures préventives adaptées.

A. Faisabilité technique

Le projet With U est jugé pleinement réalisable sur le plan technique.
Les technologies sélectionnées — React.js, Node.js, Express.js, PostgreSQL, Socket.io et YouTube IFrame API — constituent une pile cohérente et performante, adaptée à un projet collaboratif en ligne.

L'architecture adoptée est full JavaScript, garantissant une compatibilité optimale entre le front-end et le back-end.

- React.js offre une interface fluide, modulaire et réactive.
- Node.js et Express.js assurent la gestion du serveur, des API et des requêtes clients.
- Socket.io permet la communication en temps réel, nécessaire à la synchronisation vidéo et au chat collaboratif.
- PostgreSQL garantit la fiabilité, la sécurité et la cohérence des données utilisateurs.
- YouTube IFrame API facilite l'intégration du lecteur vidéo tout en respectant les standards du web et les droits de diffusion.

Les tests techniques réalisés en amont ont confirmé la stabilité de la communication client-serveur et la compatibilitéentre les composants.
Cette architecture est à la fois moderne, documentée et évolutive, ce qui la rend adaptée au niveau de complexité du projet et aux compétences de l'équipe.

B. Faisabilité organisationnelle

Sur le plan organisationnel, la faisabilité du projet repose sur une répartition claire des responsabilités et une collaboration efficace entre les membres de l'équipe.

L'équipe, composée de quatre étudiants, présente des compétences complémentaires permettant un équilibre entre les pôles front-end et back-end :

- Le front-end s'occupe de la conception de l'interface utilisateur, de l'intégration de l'API YouTube et de la gestion des interactions.
- Le back-end gère le serveur, la base de données, la sécurité et la communication temps réel via Socket.io.

Cette organisation favorise un développement parallèle et cohérent entre les différentes parties de la plateforme.

Le projet s'appuie sur des outils collaboratifs professionnels pour garantir la coordination et la traçabilité :

- Jira Software pour la planification et le suivi des tâches.
- GitHub pour la gestion du code source et le contrôle de version.
- Notion et Google Docs pour la documentation et la rédaction des livrables.

Ces outils assurent une communication fluide, une bonne visibilité de l'avancement et une répartition équilibrée du travail.

C. Faisabilité temporelle

La durée totale du projet est fixée à douze semaines, correspondant à un semestre universitaire.
Cette période a été découpée en cinq grandes phases successives :

Analyse et cadrage du besoin — 1 semaine
→ Étude du sujet, définition des objectifs et rédaction du cahier des charges.

Conception UML et architecture — 2 semaines
→ Réalisation des diagrammes UML, choix techniques et préparation du diagramme de Gantt.

Développement du MVP (Minimum Viable Product) — 6 semaines
→ Implémentation des fonctionnalités principales : salons, chat, synchronisation vidéo, authentification.

Tests et validation — 2 semaines
→ Tests unitaires, correction des anomalies, validation du MVP final.

Soutenance et livraison finale — 1 semaine
→ Rédaction du rapport, préparation de la présentation orale et démonstration du projet.

Cette planification assure une progression régulière, tout en intégrant des marges de sécurité pour compenser d'éventuels retards.
Les phases les plus critiques (intégration Socket.io et API YouTube) sont traitées en priorité pour garantir la stabilité du MVP.

D. 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 (lecture de vidéos locales) 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 travaillent sur différents systèmes (Windows, macOS).
→ Un environnement commun standardisé (Node.js, PostgreSQL, VSCode) 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.

E. 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	Prévoir un mode local de test et 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 Discord
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 conçue pour 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 développement du projet **With U** suit une **approche en cascade**, composée de plusieurs phases successives.
Chaque étape dépend de la validation de la précédente, garantissant une progression logique et contrôlée.
Cette méthode, fréquemment utilisée dans les projets académiques, favorise la clarté, la traçabilité et la structuration du travail.

Afin d'améliorer la réactivité et la communication, certains principes **agiles** ont été intégrés, tels que :

la mise en place de **sprints hebdomadaires** pour mieux répartir la charge de travail ;

des **bilans réguliers** pour suivre l'avancement et résoudre les blocages ;

et l'utilisation d'**outils collaboratifs** (Jira, GitHub, Notion) pour la coordination.

Les cinq grandes étapes du modèle appliqué au projet :

Analyse et définition du besoin

- Étude du sujet, définition des fonctionnalités essentielles du MVP.
- Identification des utilisateurs cibles et des contraintes techniques.
- Rédaction du cahier des charges et validation par l'équipe.

Conception et modélisation

- Élaboration des diagrammes UML (cas d'utilisation, classes, séquence).
- Conception de la base de données et de l'architecture logicielle.
- Création des maquettes d'interface sur **Figma**.

Développement et intégration

- Mise en place de l'environnement technique (Node.js, React, PostgreSQL).
- Implémentation progressive du MVP.
- Synchronisation entre front-end et back-end à chaque sprint.

Phase de test et validation

- Réalisation de tests unitaires, fonctionnels et d'intégration.
- Validation de la stabilité du système et correction des anomalies.

Soutenance et livraison finale

- Rédaction de la documentation finale (rapport, présentation, annexes).
- Préparation de la soutenance et démonstration du produit final.

Cette combinaison du modèle en cascade et de pratiques agiles assure à la fois **structure et flexibilité**, permettant d'adapter le projet en cas d'imprévu.

3.2 Étapes et livrables du projet

Le projet **With U** s'organise sur l'ensemble du **semestre universitaire**, soit environ **16 semaines**. Chaque phase correspond à un objectif précis et produit un livrable qui atteste de l'avancement. La validation de chaque étape conditionne le passage à la suivante.

Phase	Durée	Objectifs principaux	Livrables attendus
1. Analyse et cadrage	Semaines 1 à 2	Identifier le besoin et définir le périmètre du projet	Cahier des charges validé
2. Conception et modélisation	Semaines 3 à 5	Concevoir la structure technique et fonctionnelle	Diagrammes UML, maquettes Figma, Gantt
3. Mise en place technique	Semaine 6	Installer et configurer l'environnement de travail (serveur, base de données, GitHub)	Environnement fonctionnel et documenté
4. Développement du MVP	Semaines 7 à 12	Implémenter les fonctionnalités principales (authentification, salons, chat, synchronisation vidéo)	Code source fonctionnel (front & back)
5. Tests et validation	Semaines 13 à 14	Vérifier la stabilité du système et corriger les anomalies	Rapport de tests, version stable du MVP
6. Soutenance et documentation finale	Semaines 15 à 16	Finaliser la documentation et préparer la soutenance	Rapport final, diaporama, démonstration du projet

3.3 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 — Coordinatrice & Développeuse Front-end

- Supervise l'ensemble du projet et assure la planification sur **Jira Software**.
- Rédige le **cahier des charges** et coordonne la documentation sur **Notion**.
- Développe l'**interface utilisateur** avec **React.js** et **Tailwind CSS**.
- Réalise les **tests utilisateurs** et veille à la cohérence graphique et fonctionnelle de la plateforme.

Meriem Tekdjerad — Développeur Front-end & Intégration API

- Développe les composants dynamiques de l'application (salons, chat, formulaires, navigation).
- Intègre le **lecteur vidéo YouTube** via l'**API IFrame** et synchronise les commandes avec Socket.io.
- Collabore avec Meriem pour maintenir une cohérence visuelle et ergonomique de l'interface.
- Participe aux **tests d'intégration** et aux ajustements liés à l'expérience utilisateur.

Wissam Taleb — Développeur Back-end & Base de données

- Met en place le serveur avec **Node.js** et **Express.js**.
- Conçoit les **routes API REST**, gère les échanges entre le front-end et la base de données.
- Implémente le système de **sécurité et d'authentification JWT**.
- Administre la **base de données PostgreSQL** et garantit la fiabilité du stockage des informations.

Yanis Laftimi — Développeur Back-end & Communication temps réel

- Intègre le module Socket.io pour la communication bidirectionnelle entre le client et le serveur.
- Gère la synchronisation en temps réel des vidéos et du chat.
- Optimise la stabilité du serveur et le traitement des événements réseau.
- Collabore avec Wissam sur la gestion des requêtes et l'intégration de l'API.

4

Cette répartition claire permet à chaque membre de se concentrer sur son pôle tout en maintenant une **synergie d'équipe**essentielle à la cohérence du projet.

3.4 Suivi et communication d'équipe

La communication et le suivi sont des éléments clés de la réussite du projet.
L'équipe a adopté une stratégie de collaboration moderne, basée sur la **transparence**, la **coordination continue** et l'**utilisation d'outils professionnels**.

1.Outils utilisés

- **Jira Software** : gestion des tâches, sprints et priorisation.
- **GitHub** : hébergement du code, branches de travail, intégration continue.
- **Notion / Google Docs** : documentation, comptes rendus, stockage partagé.
- **Figma** : conception des maquettes et diagrammes UML.
- **Discord / Google Meet** : réunions hebdomadaires et échanges quotidiens.

2. Communication interne

- **Réunion hebdomadaire** : chaque semaine pour évaluer l'avancement, redéfinir les priorités et répartir les nouvelles tâches.
- **Tableau Kanban sur Jira** : suivi en temps réel des tâches (*À faire → En cours → En test → Terminé*).
- **Échanges quotidiens sur Discord** : coordination rapide et résolution des problèmes techniques.
- **Revue de code hebdomadaire** : contrôle de la qualité et validation des fonctionnalités sur GitHub.

3. Suivi de la progression

Les tâches sont planifiées par **sprints de 7 jours**.

Chaque sprint débute par une **planification** (assignation des tâches) et se clôture par un **bilan** (revue d'avancement).

Tous les comptes rendus de sprint sont archivés sur **Notion**, garantissant la traçabilité du projet.

Cette organisation favorise une gestion agile, collaborative et parfaitement adaptée à un cadre académique.

3.5 Planification du projet sur un semestre universitaire

Le projet **With U** s'étend sur l'ensemble du **premier semestre universitaire**, soit une période de **environ 16 semaines**(de septembre à janvier).
Cette durée permet d'assurer une progression maîtrisée, depuis la définition des besoins jusqu'à la présentation finale, tout en respectant les contraintes de cours et de disponibilité des membres de l'équipe.

La planification a été établie sur la base d'un **modèle en cascade**, découpé en **phases successives**, chacune aboutissant à un livrable concret et validé.
Chaque étape est planifiée sur **Jira Software**, et son avancement est suivi chaque semaine à l'aide de bilans et réunions d'équipe.

Phase	Période estimée	Durée	Objectifs principaux	Livrables attendus	Responsables
Phase 1	*début septembre*	2 semaines	Comprendre le besoin, étudier le sujet et définir les fonctionnalités principales.	Cahier des charges initial validé par l'équipe.	Meriem Tekdjerad
Phase 2	*mi-septembre → début octobre*	3 semaines	Élaborer la structure technique et fonctionnelle (UML, maquettes, base de données).	Diagrammes UML, maquettes Figma, architecture du projet, Gantt.	Malek Ghabi & Wissam Taleb
Phase 3	*début octobre*	1 semaine	Installer et configurer les outils de développement (Node.js, GitHub, PostgreSQL, React).	Environnement GitHub opérationnel, base de données initiale.	Équipe complète
Phase 4	*mi-octobre → fin novembre*	6 semaines	Implémenter les fonctionnalités principales : salons, chat, synchronisation vidéo, authentification.	Code source fonctionnel (MVP complet).	Équipe complète
Phase 5	*début décembre*	2 semaines	Tester la stabilité, corriger les anomalies et valider la cohérence des modules.	Rapport de tests, version finale intégrée et stable.	Yanis Laftimi & Meriem Tekdjerad
Phase 6	*mi à fin décembre*	2 semaines	Finaliser la documentation technique et préparer la soutenance.	Rapport final, présentation PowerPoint, vidéo de démonstration.	Équipe complète

Détails complémentaires sur la planification

A. Répartition temporelle

Durée totale : environ **4 mois (16 semaines)**.

Chaque phase comporte une **livraison intermédiaire** (maquettes, UML, MVP, tests…).

Une **réunion de suivi hebdomadaire** permet d'ajuster la progression selon l'avancement réel.

B. Suivi sur Jira Software

Les tâches sont découpées en **sprints hebdomadaires** :

- **Sprints 1–2 :** Analyse & cadrage.
- **Sprints 3–5 :** Conception & UML.
- **Sprint 6 :** Mise en place technique.
- **Sprints 7–12 :** Développement du MVP.
- **Sprints 13–14 :** Tests & validation.
- **Sprints 15–16 :** Documentation & soutenance.

Chaque sprint inclut :

une **planification en début de semaine**, une **évaluation en fin de semaine**, et un **compte rendu partagé sur Notion**.

C. Répartition de la charge de travail

Chaque membre consacre en moyenne **6 à 8 heures par semaine** au projet, incluant :

- Développement technique
- Documentation et tests
- Réunions et validations

La charge est équilibrée selon les pôles :

- **Front-end (Meriem & Malek)** : interface, design, API YouTube, tests UX.
- **Back-end (Wissam & Yanis)** : serveur, base de données, sécurité, Socket.io.

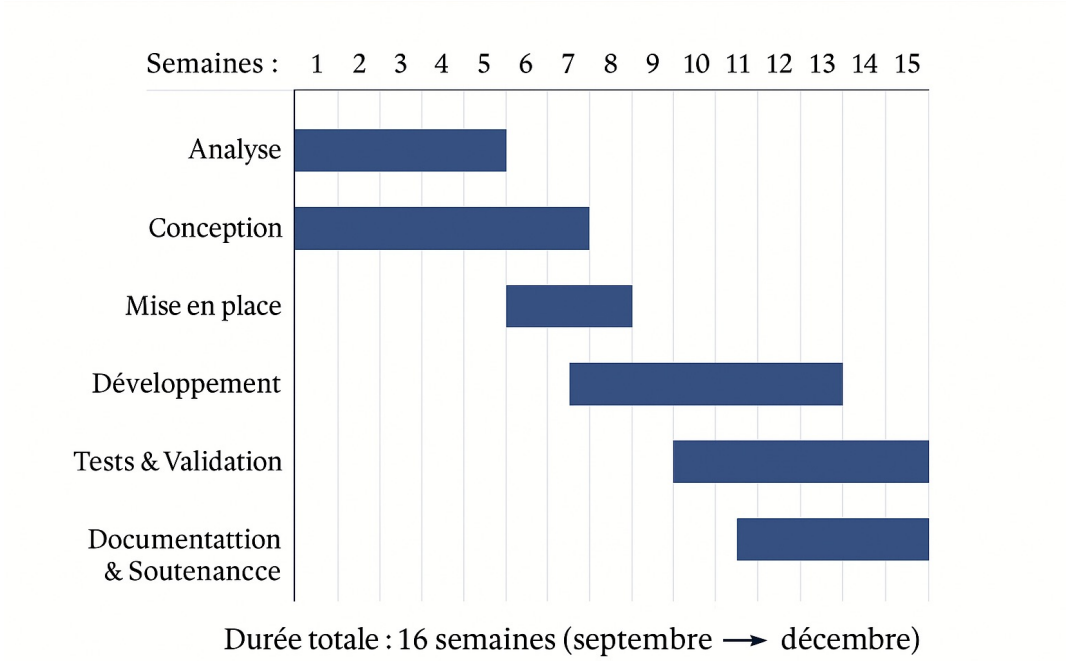
🔁 Suivi et évaluation continue

Des **bilans d'avancement** sont réalisés chaque fin de semaine sur **Jira**.

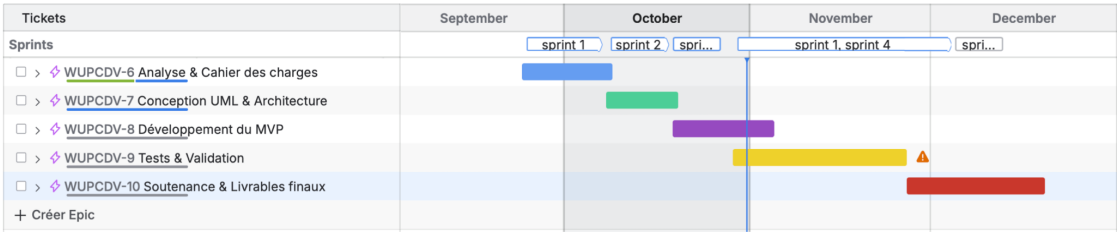
Les **réunions d'équipe** (Discord / Google Meet) permettent d'ajuster la charge et d'identifier les blocages.

Chaque tâche validée passe par un **contrôle croisé** : un autre membre vérifie le code ou le livrable.

Une **version stable du projet** doit être disponible au plus tard en **semaine 13**, laissant deux semaines pour les tests finaux et la documentation.

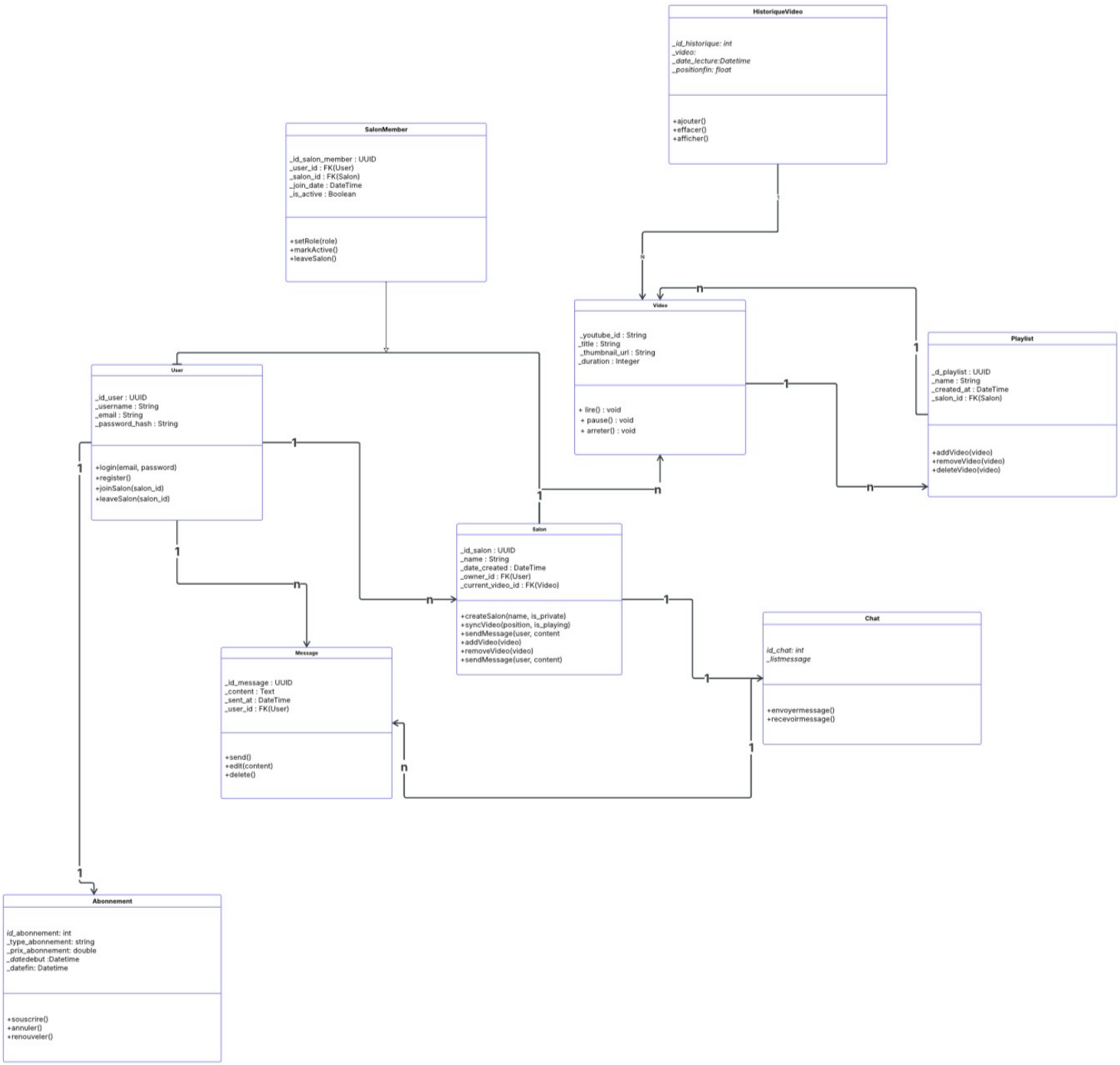


Le diagramme de Gantt détaillé figure en **Annexe C**.
Il est mis à jour régulièrement sur **Jira Software** pour refléter l'état réel du projet.

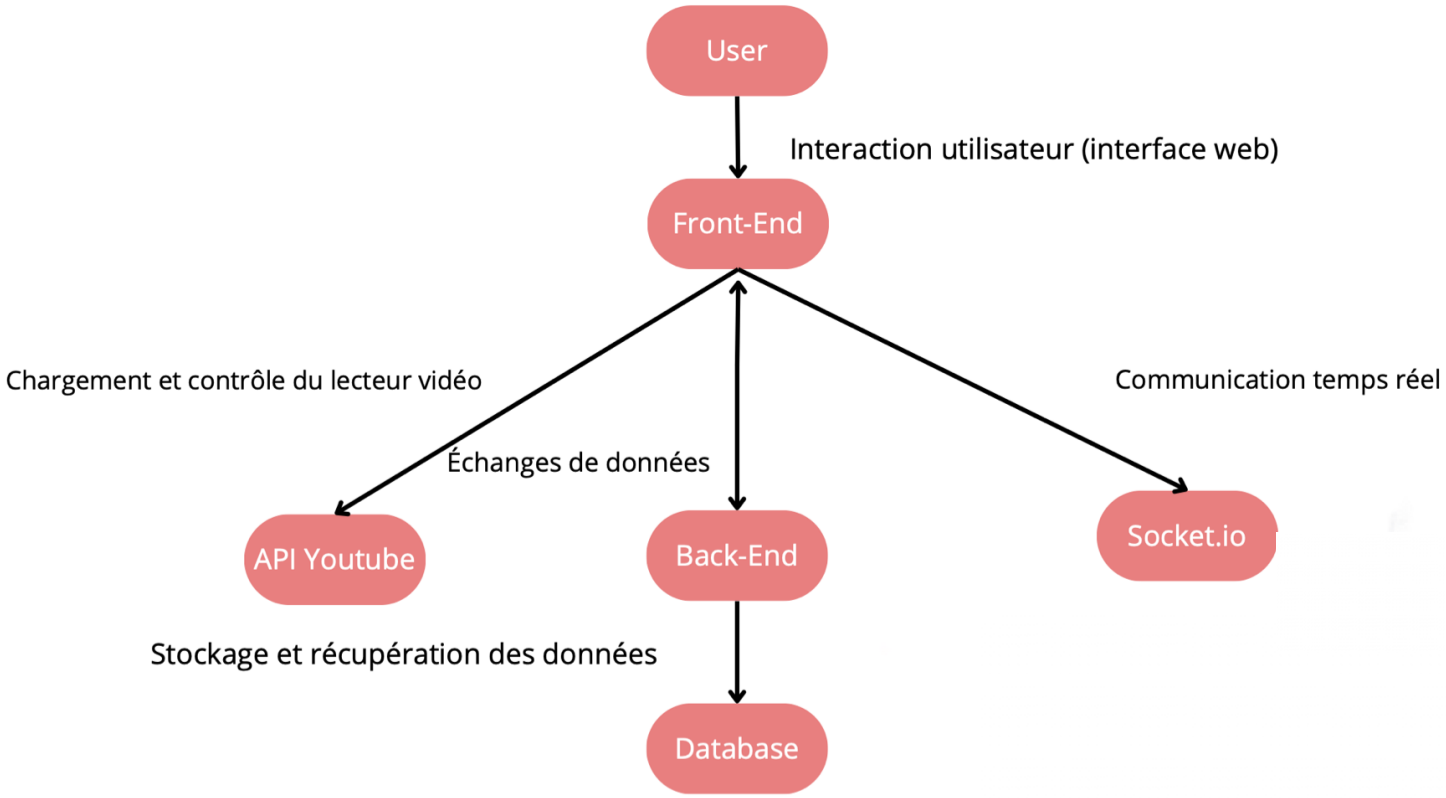


4. Annexes et visuels

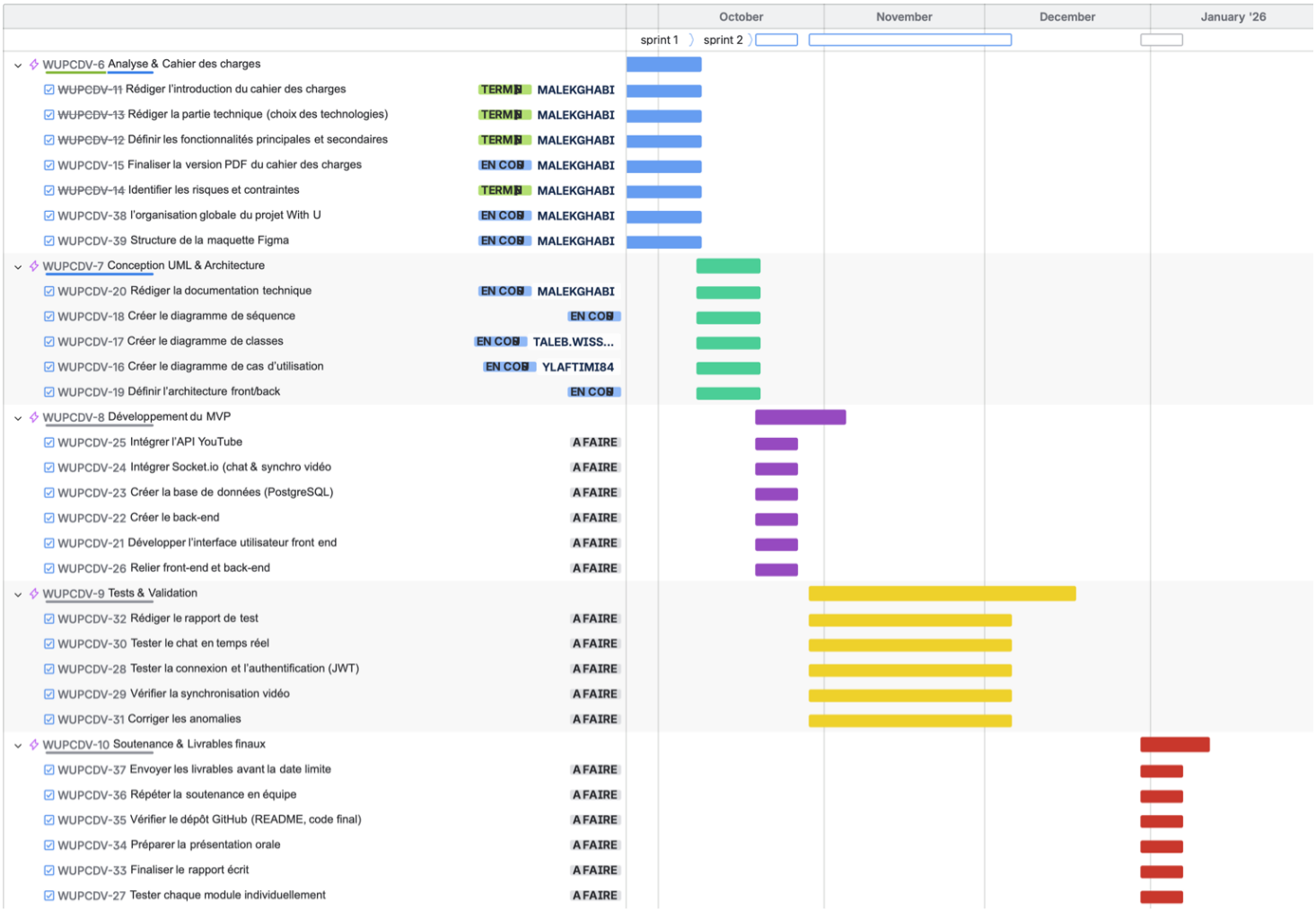
4.1 Diagramme de classes



4.2 Schéma d'architecture fonctionnelle et technique



4.3 Diagramme de Gantt



5. Conclusion

5.1 Synthèse du cahier des charges

Le présent cahier des charges fixe un **périmètre clair** pour le MVP de *With U* : objectifs, cas d'utilisation, modèle de données, architecture front/back/temps réel, critères de qualité et planification semestrielle. Les choix techniques (React, Node/Express, PostgreSQL, Socket.io, YouTube IFrame API) assurent une **cohérence full-JS**, une **maintenabilité**correcte, et une **faisabilité** démontrée au regard des compétences de l'équipe et du calendrier.

5.2 Perspectives et évolutions futures

- **Fonctionnel :** profils enrichis, rôles dans les salons, liens d'invitation, historique/"watch history", recommandations.
- **Qualité :** tests E2E (Playwright/Cypress), monitoring, logs structurés.
- **Perf & scalabilité :** rooms sharding, cache (Redis) pour états de salons, compression WS.
- **Sécurité :** rafraîchissement de tokens (refresh JWT), rate limiting, CSP renforcée.
- **Déploiement :** CI/CD GitHub Actions, hébergement managé (Render/railway), base managée (Neon/ElephantSQL).