cahier des charge

Cahier des charges – Projet IL 2025/2026 S1

Version : 3.0

Client : Enseignant

Prestataires : Lilian Serre, Sylvain Grandis, Alain Nguyen, Timeo Anchisi

---

1. Introduction

1.1 Objet du document

Ce document formalise les besoins et spécifie les fonctionnalités, contraintes et règles de développement pour la plateforme web de visionnage synchronisé de vidéos (concurrent à WatchTogether).

1.2 Portée

Livraison d'un MVP au semestre 1 comprenant :

- Création et gestion de salons.
- Lecture synchronisée de vidéos YouTube.
- Chat en temps réel.

1.3 Terminologie

- **Salon** : espace virtuel où des utilisateurs visionnent une vidéo ensemble.
- **Hôte** : créateur du salon, rôle privilégié (contrôle vidéo, modération).
- **Participant** : utilisateur invité, synchronisé à l'hôte.
- **État vidéo** : couple `{url, état lecture, timestamp}`.
- **Dérive** : décalage temporel entre clients et serveur.
- **Player Control** : outil pour contrôler l'état de lecture de la vidéo et ces options (vitesse de lecture, lecture/pause, volume (coté utilisateur), plein écran (coté utilisateur), ...)

- **Playlist** : Une liste de vidéos que le lecteur va “jouer”
- **Chat** : Bloc d’échange de messages entre participants

1.4 Abréviations

- **VIT** : Vital – requis pour MVP.
- **IMP** : Important – souhaité mais non obligatoire en S1.
- **MIN** : Mineur – amélioration ou ajout S2.
- BDD : Base De Donnée

---

2. Objectifs du produit

2.1 Définition

Application web permettant à plusieurs utilisateurs de regarder des vidéos YouTube ensemble, avec une synchronisation centralisée et un chat en temps réel.

2.2 Contexte économique

La consommation collaborative de contenu en ligne est en forte croissance. Le projet vise une alternative simple et modulaire aux solutions existantes (WatchTogether, Teleparty).

2.3 Cible

- Étudiants / jeunes adultes / famille (usage social).
- Groupes de travail (visionnage de vidéos pédagogiques).
- Associations/entreprises (visionnage en ligne).

2.4 Concurrence

- **WatchTogether** : complet mais interface lourde.
- **Teleparty** : limité à certaines plateformes.
- **SyncTube** : open source mais peu ergonomique.

2.5 Acteurs

- **Utilisateur Hôte** : crée et gère un salon.

- **Utilisateur Participant** : rejoint et suit.
- **Serveur** : maintient l'état vidéo, chat et données.
- **BDD** : stocke utilisateurs, messages, salons.

---

3. Exigences sur le produit

3.1 Exigences fonctionnelles

3.1.1 Fonctionnalités principales (VIT)

- **Créer/Rejoindre un salon**
  - Entrées : pseudo, [mot de passe salon si activé (fonctionnalité optionnel)].
  - Sorties : ID salon, code de partage, état vidéo initial, rôle utilisateur.
  - Contraintes : pseudo unique par salon, hôte unique.
  - Erreurs : pseudo déjà pris, IP bannie.
- **Lecture synchronisée YouTube**
  - Contrôleur : l'hôte.
  - Événements : `PLAY`, `PAUSE`, `SEEK`, `CHANGE_VIDEO`.
  - Tolérance dérive : max 500ms, sinon *warning latence*.
  - Nouvel utilisateur rejoint → calage automatique au timestamp hôte.
- **Chat en temps réel**
  - Entrées : pseudo, message.
  - Sorties : message diffusé à tous, avec horodatage.
  - Persistance : messages stockés en BDD tant que le serveur vit.
  - Contraintes : longueur max (200 chars), filtrage mots interdits. 2 types (censure et ban).
  - Options : flood control (MIN).

3.1.2 Fonctionnalités secondaires (IMP/MIN)

Playlist collaborative (IMP)

- **Description** : Permet à plusieurs participants d’ajouter ou de retirer des vidéos dans une liste commune que le lecteur jouera de manière séquentielle.
- **Parcours** : chaque participant peut proposer une vidéo (via URL), visible par tous dans l’ordre de lecture.
- **Entrées** : URL de la vidéo, position souhaitée (haut/bas de liste).
- **Sorties** : mise à jour de la liste visible en temps réel.
- **Contraintes** : seuls les formats compatibles avec le lecteur actif sont acceptés.
- **Priorité** : **IMP**.

Authentification utilisateur (IMP)

- **Description** : permet à un utilisateur de s’identifier via un compte enregistré pour retrouver ses salons ou paramètres personnels.
- **Parcours** : inscription (pseudo, email, mot de passe) → connexion → accès aux salons créés.
- **Entrées** : email, mot de passe (haché).
- **Sorties** : jeton d’authentification.
- **Contraintes** : stockage sécurisé des mots de passe.
- **Priorité** : **IMP**.

Historique messages permanent (MIN)

- **Description** : conservation des messages du chat même après redémarrage du serveur.
- **Parcours** : lors de la reconnexion à un salon, les messages récents sont rechargés depuis la base de données.
- **Entrées** : pseudo, message, horodatage.
- **Sorties** : historique des messages.
- **Contraintes** : durée de conservation limitée (n’est garder que x messages, historique suprimé en même temps que le salon).
- **Priorité** : **MIN**.

Protection par mot de passe du salon via URL (MIN)

- **Description** : l'hôte peut **activer** une protection par mot de passe sur un salon et **définir/retirer** le MDP.
- **Parcours** : toute tentative d'accès via **l'URL de partage** déclenche une **demande de MDP** si la protection est active.
- **Entrées** : password (côté utilisateur), enabled + newPassword|null (côté hôte).
- **Sorties** : statut d'activation, demande de MDP lors de la connexion, erreurs en cas de MDP incorrect.
- **Contraintes** : MDP stocké **haché**, **jamais renvoyé** par l'API, protection anti-bruteforce (temporisation après N échecs).
- **Erreurs** : 401 UNAUTHORIZED (MDP manquant/incorrect), 429 TOO_MANY_ATTEMPTS .
- **Priorité** : **MIN**.

3.1.3 Interopérabilité

- Fonctionne sur les principaux navigateurs (Chrome, Firefox, Safari, Edge).
- Compatible PC et mobile (responsive).

3.1.4 Conformité réglementaire

- Respect de la RGPD (aucune donnée sensible stockée). definir plus en detail
- Respect des conditions d'utilisation de l'API YouTube.
- Modération minimale du chat (filtrage mots interdits, censure et ban ip (2 types)).

3.2 Exigences non fonctionnelles

- **Fiabilité (VIT)** : reconnexion auto, resynchro immédiate.
- **Performance (VIT)** : latence max 500ms inter-clients.
- **Sécurité (IMP)** : bans d'un salon par IP (MIN), pseudos uniques, mot de passe salon (MIN).
- **Accessibilité (IMP)** : responsive, interface simple.

- **Maintenabilité (IMP)** : architecture modulaire, documentation claire.
- **Mise en place de tests unitaires (IMP) :** tests des fonctions pour s'assurer de leur bon fonctionnement

---

3.3 Exigences techniques et de réalisation

- **Front-end** : React + Tailwind + YouTube IFrame API.
- **Back-end** : Node.js + Express + Socket.IO.
- **BDD** : PostgreSQL via Prisma.
- **Déploiement** : VPS OVH Ubuntu.
- **Gestion projet** : GitHub (versioning), Notion (organisation et suivi heures)
- **Structure modulaire** :
  - Module Salon
  - Module Synchro Vidéo
  - Module Chat
  - Module Modération
  - Module Communication BDD
  - Module Playlist (optionnel)
  - Module Auth (optionnel)

3.3.1 Méthodologie de travail et organisation

Le projet suit une **méthodologie de développement inspirée du Trunk-Based Development (TBD)** :

- Le code source principal est conservé sur une branche principale (*trunk*) stable et toujours fonctionnelle.
- Chaque **nouvelle version majeure** du projet (ex : v1.0, v2.0) est développée sur une **branche dédiée**.
- Les **fonctionnalités secondaires ou correctifs** sont intégrés via de **petites branches temporaires**, fusionnées rapidement dans le tronc principal pour limiter les divergences.
- Cette approche permet de **livrer fréquemment des versions utilisables** et de garantir la **stabilité du projet** tout en favorisant la collaboration.

Les commits suivent une nomenclature stricte (feat/, fix/, refactor/, test/), et chaque merge dans main nécessite une revue de code par un autre membre de l'équipe.

Répartition des rôles

Membre	Rôle principal
Lilian Serre	Back-end, gestion serveur et base de donnée
Sylvain Grandis	Back-end
Alain Nguyen	Front-end
Tim	Front-end

Un suivi des tâches et des versions est effectué sur **Notion** et **GitHub**, avec des **réunions hebdomadaires** de synchronisation d'équipe.

3.3.2 Justification technique

Les technologies ont été sélectionnées pour garantir un **équilibre entre performance, maintenabilité et simplicité** d'implémentation :

Technologie	Justification
React.js	Permet de rafraichir la page en temps réel afin que le chat et le lecteur vidéo soit instantané et à jour.
Tailwind CSS	Un framework CSS léger et rapide pour stylisé la page et simple à utilisé.
Node.js + Express	Environnement léger et idéal pour des communications temps réel. Express offre une structure claire pour l'API REST.
Socket.IO	Bibliothèque robuste pour la synchronisation temps réel (lecture, chat) entre plusieurs clients avec faible latence.
PostgreSQL (via Prisma)	Base de données relationnelle fiable, bien adaptée à la gestion structurée des utilisateurs, salons et messages. Prisma simplifie les interactions et la validation des schémas.
VPS OVH Ubuntu	Offre un environnement flexible, stable et maîtrisé pour le déploiement. Compatible avec Node.js et PostgreSQL.

3.3.3 Risques et faisabilité

Le développement du projet repose sur des technologies connues et maîtrisées par l'équipe, rendant le MVP tout à fait réalisable dans les délais du semestre.

Cependant, plusieurs **risques techniques et organisationnels** ont été identifiés.

Chaque risque est accompagné de sa **solution de mitigation** pour en limiter l'impact.

Risque	Description	Impact	Solution / Mitigation
Latence réseau	Des différences de délai entre utilisateurs peuvent désynchroniser la lecture.	Moyen	Implémentation d'un **mécanisme de resynchronisation automatique** via Socket.IO toutes les 10 secondes, et bouton manuel "Resync".
Blocage API YouTube	L'API peut limiter les requêtes ou modifier ses conditions.	Élevé	🤷‍♂️
Perte de connexion serveur	En cas de crash serveur ou coupure réseau, le salon pourrait être interrompu.	Moyen	**Sauvegarde périodique** de l'état vidéo et du chat en BDD ; **reconnexion automatique** des clients.
Charge simultanée élevée	Plusieurs salons actifs peuvent solliciter le serveur au-delà de sa capacité.	Faible	Hébergement sur **VPS scalable**, et mise en place de **limite de sessions actives** par utilisateur.
Erreurs humaines / bugs en prod	Mauvaise manipulation de branche, merge non testé.	Moyen	Stratégie **Trunk-Based Development**, code review obligatoire, tests unitaires et CI locale avant merge.
Retard dans la coordination d'équipe	Décalage entre front et back, ou retard de livraison d'un module.	Moyen	**Réunions hebdomadaires** et jalons intermédiaires fixés
Sécurité / spam dans le chat	Utilisateurs malveillants (flood, insultes).	Faible	Filtrage des messages, **flood control**, **ban IP** temporaire.

3.4.1 Cas d'utilisation

Cas d’utilisation n°1 – Création d’un salon

- **Acteur principal :** Hôte
- **Préconditions :** L’utilisateur saisit un pseudo non pris.
- **Scénario :**
  1. L’hôte accède à la page d’accueil.
  2. Il saisit son pseudo et crée un salon.
  3. Le serveur génère un ID et un code de partage.
  4. L’hôte peut copier l’URL du salon et la partager.
- **Postconditions :** Le salon est créé et en attente de participants.

Cas d’utilisation n°2 – Lecture synchronisée d’une vidéo

- **Acteurs :** Hôte, Participants
- **Préconditions :** Les utilisateurs sont connectés dans le même salon.
- **Scénario :**
  1. L’hôte lance une vidéo YouTube.
  2. Tous les clients reçoivent le signal PLAY via Socket.IO.
  3. En cas de latence > 500ms, le client recoit une alerte et a la possibilité de se resynchronisé.
  4. L’hôte peut mettre en pause, changer de vidéo ou avancer le curseur.
- **Postconditions :** Tous les clients sont synchronisés sur le même timestamp.

Cas d’utilisation n°3 – Chat en temps réel

- **Acteurs :** Tous les participants
- **Préconditions :** Connexion au salon acltive.
- **Scénario :**
  1. L’utilisateur saisit un message et l’envoie.
  2. Le message est filtré (mots interdits) puis enregistré en BDD.
  3. Le serveur diffuse le message à tous les participants avec horodatage.

4. En cas de redémarrage du serveur, l’historique peut être rechargé.

- **Postconditions :** Le message s’affiche dans la fenêtre de chat pour tous les utilisateurs.

3.4 Structure de la page

Head : Barre de texte (qui sera utilisée pour mettre le lien de la source de la vidéo), *(optionnel : autre lien pour autre type de lecteur vidéo), bouton menu : sélection de autre type de lecteur, nom du salon)*.

Body : Le lecteur vidéo, “Player Control” en dessous du lecteur vidéo mais dans le bloc lecteur vidéo.

Carte latéral : contenant le Chat, *(optionnel : dans le bloc du chat peut avoir plusieurs fonctions (le chat) ou bien l’affichage de la playlist)*,

Footer : contenant les participants dans le salon,

Menu latéral (dans le body) : avec des options (“mention légale”, invité des participants, *(optionnel : paramètres du salon : changer le thème (de couleur proposée par le site (optionnel : l’hôte peut prendre la couleur de son choix)), nom du salon)*),

---

4. Synthèse des exigences

4.1 Fonctionnelles

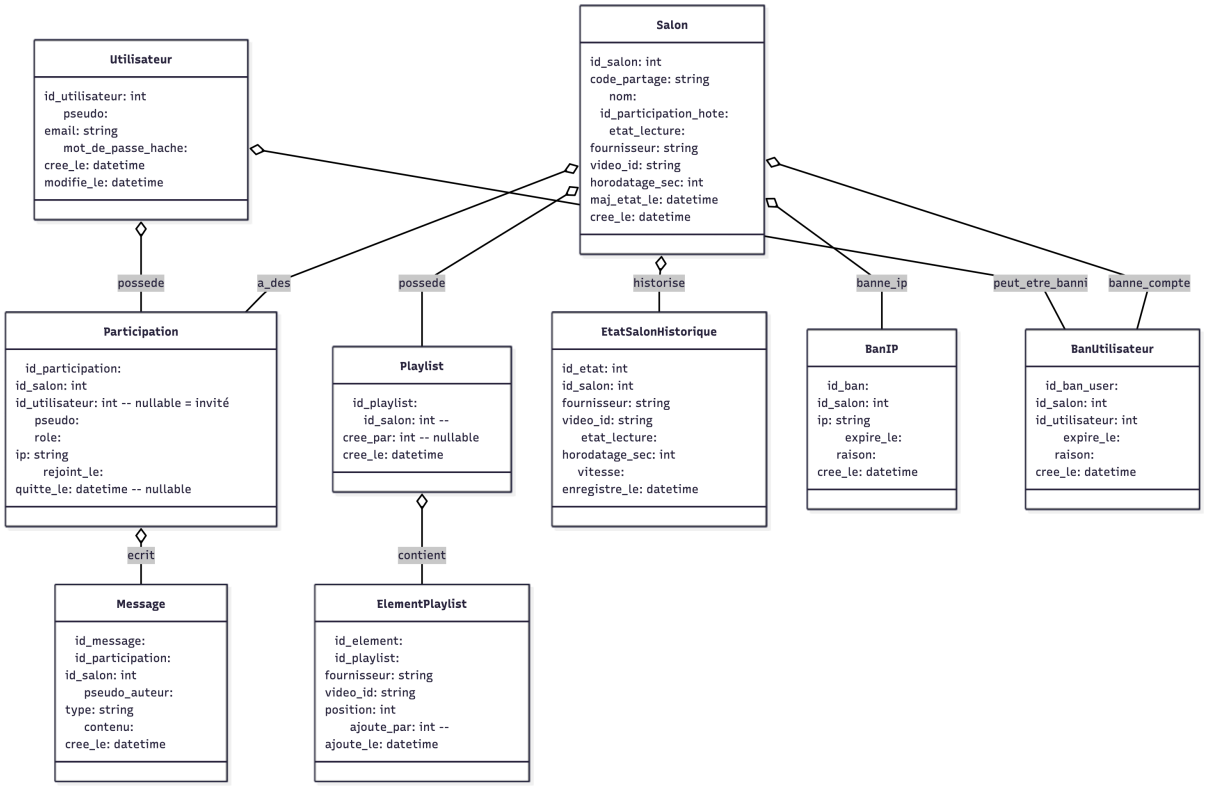
Fonctionnalité	Importance
Créer/Rejoindre salon	VIT
Lecture synchronisée YouTube	VIT
Chat temps réel	VIT

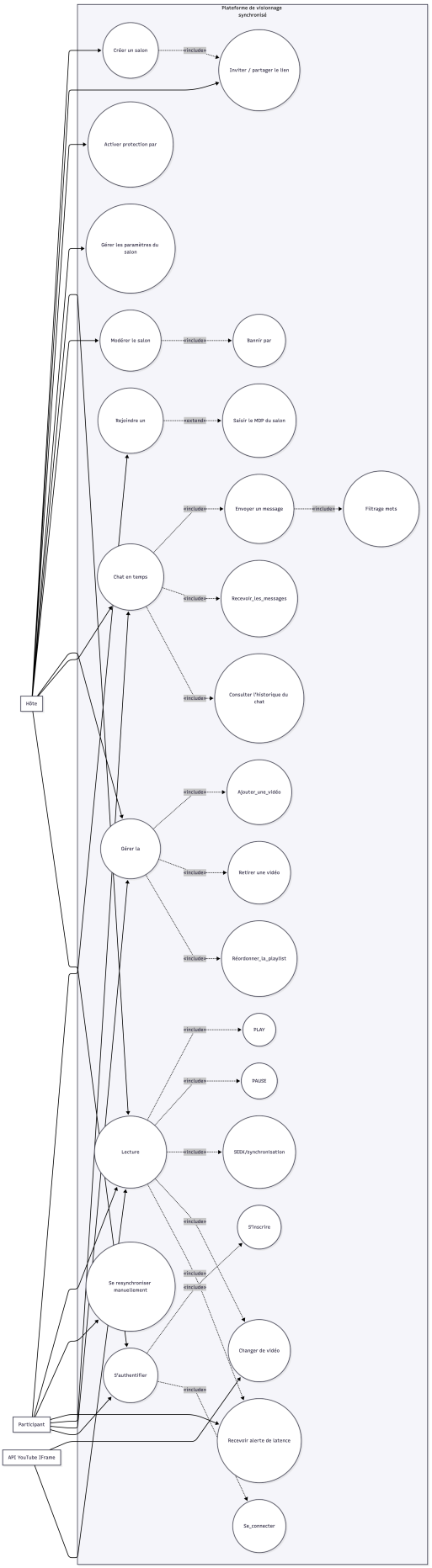
Fonctionnalité	Importance
Playlist collaborative	IMP
Authentification utilisateurs	IMP
Historique messages	MIN
Personnalisation couleur du front	MIN

4.2 Non fonctionnelles

Exigence	Importance
Fiabilité	VIT
Performance	VIT
Sécurité	IMP
Accessibilité	IMP
Maintenabilité	IMP

5. diagrammes




Plateforme de visionnage synchronisé
Créer un salon
«include»
Inviter / partager le lien
Activer protection par
Gérer les paramètres du salon
Modérer le salon
«include»
Bannir par
Rejoindre un
«extend»
Saisir le MDP du salon
«include»
Envoyer un message
«include»
Filtrage mots
Chat en temps
«include»
Recevoir_les_messages
«include»
Consulter l'historique du chat
Hôte
«include»
Ajouter_une_vidéo
Gérer la
«include»
Retirer une vidéo
«include»
Réordonner_la_playlist
«include»
PLAY
«include»
PAUSE
Lecture
«include»
SEEK/synchronisation
«include»
S'Inscrire
Se resynchroniser manuellement
«include»
«include»
Changer de vidéo
S'authentifier
«include»
Participant
Recevoir alerte de latence
API YouTube IFrame
Se_connecter