UQÀM | Département d'informatique
FACULTÉ DES SCIENCES
Université du Québec à Montréal

# Travail pratique 2 – Planification

**Présenté à :**

HAMAD, Ammar

**Dans le cadre du cours :**

INF6150 - Génie logiciel III :

Conduite de projets informatiques

Automne 2025 - Gr. 020


**Travail réalisé par :**

ÉQUIPE # 9, composée de :

**KENFACK NANDJOU, Steve Marvyn** KENS70360201

**KUBARIEVA, Oryna** KUBO65550000

**MUNYANEZA, Ursule Joana** MUNU72510007

**SOKOUDJOU SOKAMTÉ Romuald** SOKR01039701

**THIAM, Demba Aziz** THID84010203


14 octobre 2025

# Charte de projet

## Mission

Notre mission est d'améliorer et de simplifier la vie des étudiant.e.s (universitaires dans un premier temps, puis secondaire) à travers une meilleure gestion de leur temps et de leurs finances, dans le but d'améliorer leur performance académique. Nous rendons cela possible en incorporant à notre application leurs emplois du temps universitaires et profanes, et leurs budgets, pour leur offrir des programmes de révisions et de gestion de budget 100% personnalisés.

## Client

Hamad Ammar, professeur au département informatique de l'UQÀM.

## Objectifs

- Avoir un système qui authentifie les noms d'utilisateurs et les mots de passes de façon sécuritaire, permettant aux utilisateurs de se connecter sans souci à leurs comptes tout en protégeant leurs informations confidentielles.
- Avoir une base de données performante et robuste pour permettre le stockage et l'accès rapide aux données des utilisateurs par ceux-ci et l'application.
- Avoir une interface attrayante, intuitive et interactive offrant une expérience utilisateur fluide et plaisante.
- Face à des horaires chargés, des budgets serrés et de multiples échéances, UNICAL aide les étudiants à mieux répartir leur temps d'étude, suivre leurs échéances et gérer leurs dépenses.

## Portée

Inclus

- Développement d'une application web réactive et interactive permettant la gestion du temps et le suivi des finances.
- Authentification sécurisée.
- Développement d'une application mobile native (Android, IOS).
- Intégration d'une base de données ***PostgreSQL*** pour stocker les information clients.
- Rédaction d'un manuel utilisateur exhaustif.
- Intégration d'une Intelligence Artificielle (Grok ou ChatGpt) pour faire des suggestions efficaces aux utilisateurs.
- Hébergement du service sur un serveur Netlify, utilisation de XCode pour les appareils IOS et Android Studio pour les appareils basés sur Android.

- Phase de tests.
- Notifications automatiques et tableau de bord pour rappels d'événements.
- Tableau de bord administratif pour la création, modification, et suppression d'événements.
- Alerte automatique en cas de solde impayé ou échéance proche.
- Statistiques financières pour l'administration (rapport mensuels).

Exclus

- Intégration des systèmes externes non spécifiés (ex. ERP, CRM).
- Création de contenu marketing.
- Support utilisateur au-delà de la phase de déploiement initiale.
- Intégration avec des passerelles de paiement externes (ex. PayPal).
- Connexion automatique à des systèmes universitaires.
- Système multilingue (le système est livré en français seulement).

## Matrice des leviers

| | **Budget** | **Échéancier** | **Portée** |
|---|---|---|---|
| **Souple** | | | **x** |
| **Moyen** | **x** | | |
| **Rigide** | | **x** | |

## Coûts anticipés

En supposant l'équipe suivante :

- 1 Scrum Master (SM) + 1 PO (externe) + 4 développeurs = 5 personnes

Tarif horaire = 120$/hr ; Semaine de travail par membre = 15 heures-personne

Durée par itération = 3 semaines

Nombre d'itérations = 3 itérations cette session

Effort et coûts pour itérations = 3 iter. x 3 sem. x 15 hrs/sem. x 5 pers. = 675 hrs -> x 120$/hr = 81,000$ pour la phase de réalisation

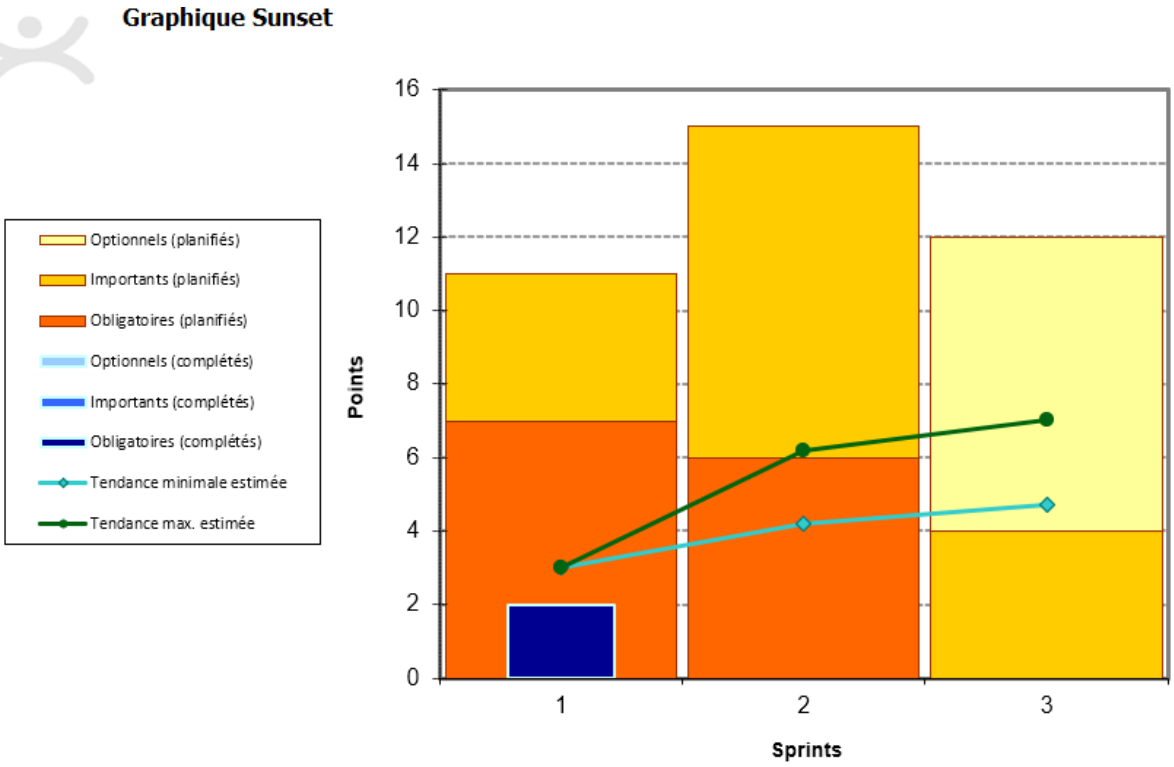
Ajouté à ça :

- La phase de **planification et de démarrage** en raison de 15hrs/sem x 5 pers. x 4 sem. -> 300 hrs x 120$/hr = 36,000$
- Et pour la phase de **clôture,** 15 hrs/sem. pour 4 semaines (1 mois) pour 3 personnes, nous avons donc 15 hrs/sem. x 4 sem. x 3 pers. -> 180 hrs x 120$/hr = 21,600$

**Nous avons donc comme total : 81,000$ + 36,000$ + 21,600$ = 138,600$**

****Note :** ces calculs sont des **ESTIMATIONS** et sont inspirés de « La technique du Sunset Graph » donné en support par le professeur HAMAD Ammar.

## Graphique Sunset :



## Critères de succès

- Le système d'authentification est sécurisé et fonctionnel
- La base de données est fiable, garantissant des résultats exacts à chaque requête et un temps de réponse rapide
- L'interface utilisateur est attrayante, intuitive et interactive
- L'IA est bien intégrée au système
- Aucune faille de sécurité ni fuite de données n'est détectée après audit
- L'application fonctionne correctement sur les principaux navigateurs (Chrome, Firefox, Edge) et sur les appareils mobiles (Android et iOS)
- Le calendrier académique et financier affiche correctement tous les événements planifiés.
- Les notifications et rappels automatiques s'envoient au bon moment
- L'intelligence artificielle (IA) est bien intégrée au système :
  - Génère des suggestions pertinentes dans au moins 80 % des cas.
- Les rapports mensuels et statistiques financières se génèrent sans erreur et reflètent fidèlement les données.
- Le manuel utilisateur est clair et permet d'utiliser l'application de manière autonome
- L'application reste accessible sur différents appareils et vitesses de connexion
- Le projet est livré dans les délais prévus
- Le budget total est respecté avec un dépassement inférieur ou égal à 10 %
- Toutes les fonctionnalités incluses dans la portée initiale sont livrées et testées avec succès

## Risques

**Risques techniques**

- **Pannes ou instabilité du serveur d'hébergement**
  1. (M) Utiliser un hébergeur fiable (AWS - backend, Netlify - hébergement statique) avec redondance et sauvegardes automatiques.
  2. (C) Basculer sur un serveur de secours et restaurer la dernière sauvegarde pour rétablir le service rapidement.
- **Défaillance de la base de données (corruption, perte ou lenteur d'accès)**
  1. (M) Mettre en place des sauvegardes automatiques quotidiennes et une surveillance de performance.
  2. (C) Restaurer la base de données à partir du dernier backup valide et effectuer un audit d'intégrité.
- **Failles de sécurité (fuite de données, piratage)**
  1. (M) Appliquer le chiffrement des données sensibles, activer HTTPS, mettre à jour les dépendances et effectuer des tests d'intrusion.
  2. (C) Suspendre temporairement les accès, corriger la faille, réinitialiser les identifiants et notifier les utilisateurs concernés.

- **Difficulté d'intégration entre les modules (authentification, calendrier, finances, IA)**
    1. (M) Définir des interfaces API claires, documentées et tester les interactions à chaque sprint.
    2. (C) Isoler le module problématique, corriger les points de friction et retester l'intégration complète.
- **Performance insuffisante de l'IA (suggestions non pertinentes)**
    1. (M) Ajuster les algorithmes et enrichir les données d'entraînement avec des cas réels.
    2. (C) Désactiver temporairement le module IA et basculer vers des recommandations statiques.
- **Incompatibilité entre versions web et mobile**
    1. (M) Effectuer des tests multi-plateformes et harmoniser les composants partagés.
    2. (C) Corriger les éléments défaillants et publier rapidement une mise à jour corrective.

**Risques fonctionnels**

- **Mauvaise interprétation des besoins fonctionnels**
    1. (M) Organiser des ateliers de clarification et valider les spécifications avec les parties prenantes.
    2. (C) Réviser les exigences non conformes et planifier un correctif fonctionnel dans le sprint suivant.
- **Bugs majeurs dans les fonctionnalités principales**
    1. (M) Appliquer des tests unitaires et d'intégration systématiques avant chaque livraison.
    2. (C) Mettre en place un plan de correction d'urgence et suspendre temporairement la fonctionnalité concernée.
- **Manque de cohérence entre tableau de bord, calendrier et rapports**
    1. (M) Centraliser la logique métier et utiliser des structures de données communes.
    2. (C) Réviser les scripts d'affichage et relancer la synchronisation des données.
- **Données incomplètes ou incorrectes saisies par les utilisateurs**
    1. (M) Ajouter des validations de champ et messages d'erreur clairs dans les formulaires.
    2. (C) Permettre la correction manuelle ou restaurer la donnée à partir des sauvegardes.
- **Problèmes de sauvegarde ou de restauration**
    1. (M) Tester périodiquement les procédures de sauvegarde/restauration.
    2. (C) Restaurer la dernière copie valide et renforcer la politique de sauvegarde.

**Risques liés à la gestion du projet**

- **Retards dans le développement**
    1. (M) Définir un calendrier réaliste avec marges de sécurité et points d'avancement hebdomadaires.
    2. (C) Réduire la portée non critique ou allonger temporairement la durée du sprint.
- **Dépassement du budget**
    1. (M) Suivre les coûts en continu et limiter les ajouts non validés.
    2. (C) Geler les nouvelles fonctionnalités et réviser le périmètre avec le client.

- **Mauvaise coordination entre membres de l'équipe**
    1. (M) Utiliser un outil de gestion (Jira, Teams, Discord) et organiser des réunions de suivi.
    2. (C) Réattribuer les tâches critiques et clarifier les responsabilités.
- **Manque de tests avant déploiement**
    1. (M) Intégrer une phase de tests obligatoires à la fin de chaque sprint.
    2. (C) Suspendre la mise en production, corriger les anomalies et relancer les tests.
- **Rotation ou absence de membres clés (ex : fonctionnalité lui a été attribuée)**
    1. (M) Documenter le code, partager les connaissances et maintenir un plan de relève.
    2. (C) Réaffecter les tâches urgentes et recruter un remplaçant temporaire.

**Risques liés aux utilisateurs et à l'adoption**

- **Résistance au changement ou manque d'engagement**
    1. (M) Impliquer les utilisateurs finaux dès la phase de conception et organiser des formations.
    2. (C) Mettre en place des séances d'accompagnement post-déploiement.
- **Interface peu intuitive**
    1. (M) Réaliser des tests UX avec un échantillon d'utilisateurs avant le déploiement.
    2. (C) Apporter des améliorations dans une version mineure (v1.1).

**Risques externes**

- **Coupures d'électricité, pannes réseau, accès Internet limité**
    1. (M) Choisir un hébergeur disposant d'une redondance géographique et d'un plan de continuité.
    2. (C) Basculer le service sur un serveur secondaire ou notifier les utilisateurs de la coupure planifiée.
- **Modifications réglementaires (RGPD, Loi 25, etc.)**
    1. (M) Surveiller les évolutions légales et adapter la politique de confidentialité.
    2. (C) Mettre à jour les paramètres de conformité et informer les utilisateurs des changements.
- **Dépendance à un fournisseur tiers (hébergeur, API, IA)**
    1. (M) Sélectionner des fournisseurs avec SLA (Service Level Agreement) clair et solution de rechange possible.
    2. (C) Migrer vers une solution alternative en cas d'interruption ou d'augmentation de coûts.

# Charte d'équipe

### Nom de l'équipe

- Notre équipe se nomme ***Alphabits*** qui combine "Alpha", symbole de nouveauté et de leadership, et "Bits", unité de base de l'informatique, pour représenter un groupe innovant à la pointe du numérique.

### Membres de l'équipe

- Demba Aziz Thiam
- Oryna Kubarieva
- Romuald Sokoudjou Sokamté
- Steve Marvyn Kenfack Nandjou
- Ursule Joana Munyaneza

## Contraintes

- L'inflexibilité de la date de remise dans ce contexte scolaire.
- L'équipe est composée de 5 développeurs, sans possibilité de recrutement supplémentaire.
- Revue obligatoire des livrables par le superviseur/enseignant avant tout déploiement.
- Chaque membre consacre un nombre d'heures limité par semaine au projet, en parallèle de ses autres activités
- Planifier les sprints, gérer le backlog et assurer le suivi des progrès

## Rôles et responsabilités

| | | |
|---|---|---|
| Développeur | - Écrire du code simple pour tester des idées, utiliser les frameworks pour améliorer la visibilité de l'application<br>- Vérifier que tout fonctionne bien, corriger les erreurs au besoin<br>- Travailler avec l'équipe pour combiner les idées et améliorer le code dans les différents cas erreurs rapidement<br>- Participer à des réunions pour valider avancement du projet et régler les problèmes | Tous les membres de l'équipe |
| Coordonnateur | - S'assurer que les autres membres de l'équipe effectuent leurs tâches et aider dans les cas des difficultés<br>- Organiser des appels ou des rencontrer courte pour discuter de l'avancement du projet et répondre aux questions si cela et nécessaire | Oryna Kubarieva |
| Assurance qualité | - Tester les idées pas à pas<br>- Chercher les fautes dans le code ou dans l'exécution de l'application<br>- Vérifier que le travail suit les règles du cours<br>- Donner les retours à l'équipe pour pouvoir travailler sur les taches à améliorer<br>- Suivre et gérer les risques | Joana Munyaneza<br>Romuald Sokoudjou Sokamté |
| Gestionnaire de la configuration | - Suit les changements<br>- Garde le code source dans Git et assurer le fonctionnement<br>- Évite les pertes en sauvegardant tout, comme les versions des séquences.<br>- Aide à retrouver les anciennes versions si besoin | Steve Marvyn Kenfack Nandjou |
| Point de contact technique | - Répond aux questions techniques, comme réparer un bug dans un diagramme<br>- Explique les outils simplement, par exemple comment utiliser<br>- Résout les problèmes rapides, comme une erreur dans une séquence<br>- Partage des astuces pour mieux modéliser, | Demba Aziz Thiam |
| Product owner | - Décide ce que le projet doit faire<br>- Priorise les tâches qui doivent être changer | Ammar Hamad, Ph.D. |

## Normes de fonctionnement de l’équipe

## a) Processus de développement choisi et justification

L’équipe adopte un processus de développement Agile inspiré de la méthode Scrum, adapté à un contexte académique. Ce choix se justifie par la nécessité d’itérer rapidement, de livrer des fonctionnalités opérationnelles à chaque sprint et de s’adapter aux retours du professeur et des utilisateurs (étudiants, enseignants, administrateurs).

Structure du processus :

- Phase d’analyse et de conception : compréhension des besoins, définition de la portée et conception de l’architecture logicielle.
- Développement incrémental : sprints de 2 à 3 semaines avec planification, développement, revue de code et tests.
- Phase de test et intégration finale : intégration complète, tests fonctionnels et vérification de la conformité à la définition de terminé.
- Livraison et documentation finale : remise du code source, manuel utilisateur et rapport technique.

Définition de « Terminé » :

- Le code est développé, testé et intégré dans GitLab.
- Aucune erreur critique n’est détectée.
- Le livrable est documenté.
- Le fonctionnement est validé par l’équipe et le professeur.

## b) Outils utilisés et leurs usages

| Outil | Usage principal | Format de sortie / transmission |
|---|---|---|
| Ionic Framework | Développement multiplateforme (web + mobile Android/iOS) | Code source GitLab (.ts, .html, .css) |
| PostgreSQL (cloud) | Base de données principale (utilisateurs, cours, échéances, blocs d'étude, dépenses) | Connexion DB (URI), exports SQL/CSV |
| GitLab | Gestion du code source, versionnement, revue de code | Lien de dépôt GitLab |
| Atlassian Confluence | Documentation et suivi collaboratif | Export PDF ou Word |
| Microsoft Word | Rédaction de tout fichier texte | Fichiers .docx ou .pdf |
| PowerPoint | Présentations de sprints | Fichiers .pptx ou .pdf |
| Microsoft Teams | Réunions en ligne | Compte rendu Word ou Confluence |

## c) Logistique de coordination de l'équipe

- Réunions hebdomadaires chaque mardi, vendredi et dimanche à 22h30 via Teams, présence obligatoire.
- Canaux de communication : Discord pour la coordination rapide, courriel pour les échanges formels, GitLab pour l'hébergement du développement code source, Confluence pour la documentation du progrès
- Tous les fichiers sont centralisés sur GitLab et Confluence, avec sauvegarde hebdomadaire par le Coordonnateur.

## d) Modes décisionnels

- Mode principal : Unanimité -> tous les membres sont d'accords et aucun n'émet d'opposition. Les décisions sont consignées dans Confluence avec la date.

- Mode secondaire : Consensus -> la majorité est d'accord et aucun membre n'exprime d'opposition forte

## e) Les comportements attendus

### Engagement et responsabilité

- Chaque membre respecte les échéances convenues et assume pleinement ses tâches
- Les membres préviennent l'équipe en cas d'empêchement ou de difficulté
- Les livrables sont remis complets, testés et conformes à la « définition de terminé »
- Chacun est responsable de la qualité de son travail et de sa contribution collective

### Communication et collaboration

- Les échanges se font dans un **esprit d'ouverture et de respect**.
- Toute question, idée ou désaccord est discuté de façon constructive.
- Les membres participent activement aux réunions et répondent aux messages dans un délai raisonnable.
- Les décisions prises en équipe sont respectées par tous.

### Esprit d'équipe

- Les succès sont collectifs : on reconnaît les efforts des autres.
- L'entraide prime sur la compétition individuelle.
- En cas de problème l'équipe cherche ensemble des solutions.
- Les nouveaux outils ou méthodes sont partagés pour favoriser la montée en compétence commune.

### Transparence et rigueur

- Le suivi des tâches est mis à jour régulièrement sur GitLab ou Atlassian.
- Les problèmes sont signalés rapidement pour éviter les blocages.
- Les décisions et changements sont documentés dans Confluence pour assurer la traçabilité.

### Respect du cadre de travail

- Les réunions commencent et se terminent à l'heure prévue.
- Les canaux de communication (Discord, GitLab, courriel, etc.) sont utilisés selon leur objectif.
- Les remarques ou critiques se font de manière professionnelle et orientée vers l'amélioration.
- Chacun veille à maintenir un climat de confiance, d'écoute et de bienveillance.

## f) Les comportement non tolérés

### Manque de respect et incivilité

- Les propos offensants, moqueurs, discriminatoires ou condescendants.
- Les attitudes agressives, autoritaires ou dénigrantes envers un membre de l'équipe.
- Les jugements personnels ou attaques directes lors des discussions techniques ou de réunions.
- Toute forme de harcèlement moral, sexuel, religieux, ou discriminatoire (âge, genre, origine, croyances, orientation, etc.).

**Manque de collaboration**

- Refuser de partager des informations ou des ressources utiles à l'équipe.
- Ignorer les messages ou les demandes de suivi sans justification.
- Travailler en silo sans tenir compte des interdépendances entre les modules.
- Se désengager des réunions ou des discussions d'équipe de manière répétée.

**Manque de professionnalisme**

- Ne pas respecter les échéances sans prévenir ni proposer de solution.
- Livrer du code ou des documents non testés, incomplets ou non conformes aux standards convenus.
- Modifier le travail d'un collègue sans l'en informer.
- Supprimer, altérer ou cacher des données ou du code du dépôt GitLab.
- Ne pas documenter son travail, ce qui nuit à la traçabilité collective.

## Résolution de conflits

- En cas de manque de respect ou d'incivilité d'un membre, l'équipe doit se réunir pour régler le problème. Le membre fautif reçoit un avertissement. À l'accumulation de trois avertissements, le membre se voit expulsé de l'équipe.
- Dans le cas du manque de participation d'un membre dans une phase, il peut se rattraper à la prochaine. S'il ne se rattrape pas, les autres membres évalueront la proportion de son travail et dans le cas où la contribution du membre est considérablement inférieure à celle des autres membres, sa note sera réduite de la moitié du pourcentage représenté par cette partie.