

UNICAL
RAPPORT DU SPRINT 1

Présenté à :

Gnagnely Serge Dogny

Dans le cadre du cours :

INM5151- Projet d'analyse et de modélisation:

Automne 2025 - Gr. 020

Travail réalisé par :

ÉQUIPE # 3, composée de :

KENFACK NANDJOU, Steve Marvyn

KUBARIEVA, Oryna

MUNYANEZA, Ursule Joana

SOKOUDJOU SOKAMTÉ Romuald

THIAM, Demba Aziz

Dimanche 9 novembre 2025

TABLE DES MATIÈRES

TABLE DES MATIÈRES	2
1. INTRODUCTION.....	3
1.1 Objectifs	4
1.2 Vue d'ensemble du produit.....	4
Description générale.....	4
Limites et bordures.....	4
Contexte et objectifs.....	5
1.3 Définitions, acronymes et abréviations.....	5
1.4 Documents de références (reprendre le contenu du ConOps)	5
2. DESCRIPTION GÉNÉRALE DU LOGICIEL	6
2.1 Vue d'ensemble des fonctions du produit	6
1. Gestion du calendrier et des études	6
2. Gestion des notifications	6
3. Gestion du budget.....	7
4. Module d'intelligence artificielle (IA – version 1)	7
5. Gestion des comptes utilisateurs	7
6. Plateforme de cours (intégration manuelle)	7
2.2 Contenu des sprints	7
Sprint 1 – Authentification et calendrier	7
Sprint 2 – Discipline, finances et IA de base	8
Sprint 3 – IA avancée et intégration complète	9
Le dernier sprint portait sur l'enrichissement des fonctionnalités d'IA et sur l'intégration globale avec des tests.	
Les développements prévus comprenaient :	9
3. CONTENU DU SPRINT ACTUEL	9
3.1 Cas d'utilisation implantés dans le sprint.....	9
<i>Cas d'utilisation 1 – Authentification de l'utilisateur.....</i>	<i>9</i>
<i>Cas d'utilisation 2 – Ajout d'une échéance au calendrier.....</i>	<i>10</i>
3.2 Modèle de classes	12
4. PROCHAIN SPRINT (1 À 2 PAGES, MAX).....	15
4.1 Revue du sprint	15
4.1.1 Répartition des tâches	15
4.1.2 Revue technique.....	16
Fonctionnalités complétées	16

Fonctionnalités partiellement complétées ou non terminées	16
Ce qui a bien fonctionné	16
Ce qui a moins bien fonctionné.....	16
Obstacles rencontrés et solutions adoptées	17
4.1.3 Revue gestion	17
Évaluation des personnes et relations	17
Évaluation des processus.....	17
Évaluation des outils	17
Points positifs	18
Points à améliorer.....	18
Plan d'action pour le Sprint suivant	18
4.2 Plan pour prochain sprint.....	18
Cas d'utilisation prévus pour le Sprint 2 :.....	18
5. ANNEXES.....	19

1. INTRODUCTION

Ce document présente une vue d'ensemble du projet UNICAL. Il débute par une introduction du système, accompagnée d'une description générale qui expose le public cible, les contraintes ainsi que le contexte d'utilisation de l'application.

Par la suite, le document aborde la présentation du Sprint 1, qui avait été préalablement planifié avant le développement. Cette section inclut la comparaison entre les estimations planifiées et les résultats réels, le contenu développé, ainsi que les fonctionnalités réalisées, tout en mettant en évidence les contraintes et les difficultés rencontrées par l'équipe durant le processus de développement.

Les cas d'utilisation sont ensuite présentés afin d'illustrer le fonctionnement du système dans sa version actuelle et de montrer concrètement comment les différentes fonctionnalités peuvent être utilisées par les utilisateurs.

Enfin, le document se conclut par une brève présentation du prochain sprint, décrivant les estimations prévues, les fonctionnalités qui seront ajoutées ou améliorées, ainsi que les objectifs à atteindre lors de la prochaine phase de développement.

1.1 Objectifs

L'objectif de ce document est de présenter l'ensemble des fonctionnalités préétablies qui ont été réalisées au cours du développement du Sprint 1, tout en décrivant les démarches entreprises ainsi que les difficultés rencontrées par chaque membre de l'équipe. De plus, il offre un premier aperçu du logiciel à travers son interface utilisateur, ses principales pages, et le fonctionnement général des modules déjà implémentés.

1.2 Vue d'ensemble du produit

Description générale

UNICAL est une application conçue pour centraliser la gestion du temps et du budget des étudiants. Elle regroupe dans un même espace :

- Le calendrier académique (cours, travaux, examens),
- Les créneaux d'étude personnalisables (minuteur de 25, 45, 60 ou 90 minutes, avec pause et reprise),
- Et un outil de gestion budgétaire simplifié (suivi des revenus et dépenses, alertes en cas de dépassement).
- L'application intègre une intelligence artificielle (IA) qui :
- Propose des priorités de base (version initiale),
- Effectue une replanification automatique en cas de conflit dans l'emploi du temps,
- Fournit des repères budgétaires simples pour aider à mieux gérer les finances étudiantes.

UNICAL offre également des vues synthétiques (par semaine, « prévu vs réalisé », barres de progression) ainsi que des notifications cohérentes pour les échéances académiques et les rappels budgétaires.

Limites et bordures

Le système :

- Ne remplace pas les plateformes institutionnelles officielles (ex. Moodle, Omnivox, Teams, etc.),
- Ne gère pas les communications entre étudiants ou enseignants,
- Ne vise pas la comptabilité avancée ni la gestion de dettes complexes.

Il agit comme un complément intelligent et pratique à la planification académique et financière.

Contexte et objectifs

UNICAL s'inscrit dans un contexte où de nombreux étudiants doivent jongler entre études, projets et emploi, tout en respectant un budget limité.

Les objectifs principaux du projet sont :

- d'optimiser la gestion du temps et du budget au quotidien,
- de réduire la charge mentale liée à la planification,
- et de favoriser l'autonomie et la productivité des étudiants.
- Les bénéfices attendus incluent :
- une meilleure organisation académique,
- une vision claire du progrès personnel,
- et une meilleure maîtrise des finances étudiantes.

Les retombées du projet visent à améliorer la réussite académique et le bien-être global des étudiants.

1.3 Définitions, acronymes et abréviations

Non applicable

1.4 Documents de références (reprendre le contenu du ConOps)

Voici la liste des applications qui ont été utilisé comme la source de l'inspiration pour notre système :

- Pomodoro Tracker

Pomodoro Tracker. Pomodoro Tracker (Version web) [Application web de gestion du temps]. Pomodoro-tracker.com. <https://pomodoro-tracker.com/>

- YEolPumTa (YPT)

YeolPumTa. YeolPumTa (Version mobile) [Application de productivité et d'étude]. YeolPumTa. (Disponible sur Google Play et App Store)

- Todoist

Doist. Todoist (Version web et mobile) [Application de gestion de tâches]. Doist Ltd. <https://www.todoist.com/>

- Notion

Notion Labs Inc. Notion (Version web et mobile) [Application de prise de notes et de gestion de projets].
Notion Labs Inc. <https://www.notion.com/>

- Google Tasks

Google LLC. Google Tasks (Version web) [Application de gestion de tâches]. Google Workspace.
<https://workspace.google.com/products/tasks/>

- Trello

Atlassian. Trello (Version web et mobile) [Application de gestion de projets et d'organisation visuelle].
Atlassian. <https://trello.com/>

- Clockify

COING Inc. Clockify (Version web et mobile) [Application de suivi du temps et de productivité]. COING Inc. <https://clockify.me/>

- Microsoft To Do

Microsoft Corporation. Microsoft To Do (Version web et mobile) [Application de gestion de tâches et rappels]. Microsoft 365. <https://to-do.microsoft.com/>

2. DESCRIPTION GÉNÉRALE DU LOGICIEL

2.1 Vue d'ensemble des fonctions du produit

Le système UNICAL est une application web et mobile destinée aux étudiant·e·s souhaitant planifier efficacement leur temps d'étude, suivre leur progression et gérer leur budget personnel. Les principales fonctionnalités sont regroupées selon les modules suivants :

1. Gestion du calendrier et des études

- Créer et modifier des échéances : l'utilisateur peut ajouter, modifier ou supprimer des travaux, examens ou cours avec des dates précises.
- Génération automatique de blocs d'étude : l'application propose des blocs prédéfinis (25, 45, 60 ou 90 minutes) associés à chaque échéance.
- Minuteur intégré : permet de démarrer, mettre en pause et reprendre les séances d'étude avec un retour visuel sur le temps écoulé.
- Suivi de la progression : l'utilisateur peut comparer ce qui était prévu avec ce qui a réellement été accompli (« prévu vs réalisé »).

2. Gestion des notifications

- Alertes d'échéances : rappels automatiques avant les dates limites ou débuts de séances d'étude pour qu'utilisateur peuvent être à jour dans ses études ou les planifications.
- Notifications budgétaires : Envoi d'alertes lorsque le budget dépasse un seuil défini. Cette notification peut être entièrement activée ou désactivée par l'utilisateur. Une suggestion indique toutefois qu'il est important de la conserver afin que l'utilisateur puisse être informé de ses dépassements et s'ajuster au besoin pour les prochaines fois.

- Préférences personnalisées : chaque utilisateur peut ajuster la fréquence et le type de notifications reçues

3. Gestion du budget

- Saisie des revenus et dépenses : interface simple pour enregistrer les mouvements financiers.
- Suivi du solde : affichage du budget restant et des tendances de dépense.
- Alerte de dépassement : avertissement automatique si le budget prévu est dépassé.

4. Module d'intelligence artificielle (IA – version 1)

- Évaluation de la priorité : classement basique des tâches selon leur importance et leur date d'échéance.
- Replanification proposée : suggestions automatiques de réorganisation en cas d'imprévu, sans action obligatoire de l'utilisateur (approche non prescriptive).

5. Gestion des comptes utilisateurs

- Création et authentification de comptes : chaque utilisateur dispose d'un compte personnel protégé par mot de passe en utilisant son courriel personnel comme la mode de la connexion et la communication entre la base de données et l'authentification
- Sauvegarde et récupération de données : toutes les informations sont stockées de manière sécurisée et sauvegardées régulièrement dans la base de données qui est connecté via API. La base de donnée en question se retrouve en ligne, alors elle n'est pas directement implémentée dans le code source du projet.

6. Plateforme de cours (intégration manuelle)

- Consultation externe : l'étudiant consulte sa plateforme universitaire habituelle pour connaître ses cours.
- Saisie manuelle dans UNICAL : il peut ensuite ajouter les cours et échéances dans l'application pour planifier son emploi du temps, les cours seront sauvegardés dans la base de données pour faciliter leur manipulation dans les différentes parties du calendrier.

2.2 Contenu des sprints

Sprint 1 – Authentification et calendrier

Concernant la planification initiale du **Sprint 1**, les fonctionnalités suivantes ont été prises en compte pour être développées :

1. Authentification
2. Création du profil utilisateur
3. Vue du calendrier
4. Personnalisation du calendrier
5. Ajout des échéances (ex. : quiz, examens, travaux, etc.)
6. Modification des échéances
7. Système de notifications
8. Système de gestion de la salle de passage

Cependant, certaines de ces fonctionnalités n'ont pas pu être entièrement développées, car plusieurs difficultés ont été rencontrées, ce qui a nécessité plus de temps pour les résoudre et les corriger.

La **page d'authentification** est développée et fonctionnelle, mais de petits messages d'erreur s'affichent encore, même lorsque tous les champs sont correctement remplis. Il sera donc essentiel de corriger ce comportement afin d'améliorer l'expérience utilisateur.

La **page de création du profil** fonctionne correctement. Elle permet à l'utilisateur de saisir ses informations personnelles, telles que le nom, le prénom, le courriel et le mot de passe, afin de sécuriser l'accès à son compte.

Les **fonctionnalités liées au calendrier** sont également opérationnelles. Le calendrier s'affiche dans différents modes et peut être personnalisé selon les préférences de l'utilisateur.

Les **fonctionnalités liées aux échéances** fonctionnent aussi correctement : il est possible d'ajouter des examens, des travaux pratiques ou d'autres tâches, en précisant l'heure, la date, la salle et d'autres informations pertinentes.

En revanche, le **système de notifications** est encore en phase de développement. Certaines difficultés ont été rencontrées lors de la fusion des branches de code nécessaires pour tester et valider son bon fonctionnement.

L'ensemble des fonctionnalités développées, avec les estimations et les heures réelles de travail, est présenté dans le document **Product Backlog** en annexe.

Sprint 2 – Discipline, finances et IA de base

Le deuxième sprint s'est concentré sur les modules de discipline d'étude, de gestion budgétaire et sur une première version de l'intelligence artificielle. Il inclutait :

- Le calcul de priorité des événements et la barre de progression du travail réalisé.
- Le placement automatique de blocs d'étude et la vue quotidienne « Plan d'étude ».

- Le chronomètre d'étude (démarrage, pause, reprise, fin de session, prolongation).
- Le suivi de progression académique par graphiques.
- La gestion complète des revenus et dépenses, le suivi budgétaire mensuel et les alertes en cas de dépassement de budget.

Sprint 3 – IA avancée et intégration complète

Le dernier sprint portait sur l'enrichissement des fonctionnalités d'IA et sur l'intégration globale avec des tests.

Les développements prévus comprenaient :

- La re-priorisation intelligente des tâches et l'apprentissage adaptatif (ajustement des durées selon la performance).
- Le placement optimal des blocs d'étude en fonction des contraintes personnelles (emploi, fatigue, etc.).
- L'ajout de conseils de bien-être, de notifications motivationnelles et de questions de suivi personnel.
- L'analyse avancée des finances par l'IA et la génération de rapports globaux de fin de semestre, combinant progression académique, situation financière et recommandations personnalisées.

3. CONTENU DU SPRINT ACTUEL

3.1 Cas d'utilisation implantés dans le sprint

Cas d'utilisation 1 – Authentification de l'utilisateur

Les acteurs principaux : les utilisateurs (étudiants)

Brève description : Ce cas d'utilisation décrit le processus par lequel un nouvel utilisateur crée un profil personnel afin d'accéder à la plateforme. Le profil comprend des informations de base nécessaires à l'authentification et à la personnalisation du compte.

Préconditions :

- L'utilisateur a ouvert l'application et se trouve sur la page d'accueil ou de connexion.
- Le système est accessible et opérationnel.

Postconditions :

- Le profil de l'utilisateur est créé et enregistré dans la base de données.
- L'utilisateur peut se connecter à son compte à l'aide de ses identifiants.

Scénario principale :

1. L'utilisateur clique sur Créer un compte depuis la page d'accueil.
2. Le système affiche le formulaire d'inscription.
3. L'utilisateur saisit ses informations personnelles :
 - a. Prénom
 - b. Nom
 - c. Adresse courriel
 - d. Mot de passe
4. L'utilisateur clique sur le bouton S'inscrire.
5. Le système vérifie la validité des champs saisis (format du courriel, longueur du mot de passe, etc.).
6. Le système enregistre les informations de l'utilisateur dans la base de données.
7. Le système confirme la création du compte et redirige l'utilisateur vers la page de connexion à l'utilisateur
8. L'utilisateur peut désormais se connecter avec ses nouvelles informations.

Scénarios alternatifs :

Champ obligatoire manquant :

1. Si un champ obligatoire n'est pas rempli
2. Le système affiche le message : « Veuillez remplir tous les champs obligatoires. ».

3. Le scénario reprend à l'étape 2.

Courriel déjà existant :

1. Si le courriel entré existe déjà dans la base de données,
2. Le système affiche le message : « Ce courriel est déjà associé à un compte existant. ».
3. Le scénario s'arrête.

Erreur d'enregistrement :

1. Si une erreur survient pendant l'enregistrement,
2. Le système affiche le message : « Erreur : impossible de créer le profil pour le moment. ».
3. Le scénario s'arrête.

Cas d'utilisation 2 – Ajout d'une échéance au calendrier

Les acteurs principaux : les utilisateurs (étudiants)

Brève description : Ce cas d'utilisation décrit le processus par lequel un utilisateur ajoute une nouvelle échéance (examen, travail, quiz, etc.) dans son calendrier afin de planifier ses activités d'étude.

Préconditions :

- L'utilisateur est connecté à son compte.
- Le calendrier personnel est accessible.

Postconditions :

- L'échéance est enregistrée dans le système.
- L'utilisateur peut visualiser cette échéance dans son calendrier et la modifier au besoin.

Scénario principal :

1. L'utilisateur accède à la vue du calendrier.
2. Le système affiche le calendrier avec les échéances existantes.
3. L'utilisateur clique sur le bouton Ajouter une échéance.
4. Le système affiche le formulaire d'ajout d'échéance.
5. L'utilisateur saisit les informations suivantes :
 - a. Titre de l'échéance
 - b. Date et heure
 - c. Type d'échéances
 - d. Notes supplémentaires
6. L'utilisateur clique sur le bouton Enregistrer.
7. Le système vérifie la validité des données saisies.
8. Le système enregistre l'échéance dans la base de données.
9. Le système met à jour l'affichage du calendrier pour inclure la nouvelle échéance.

Scénarios alternatifs :

Champs manquants :

1. Si un champ obligatoire (ex. titre ou date) est manquant
2. Le système affiche le message : « Veuillez remplir tous les champs obligatoires. »
3. Le scénario s'arrête

Erreur lors de l'enregistrement :

1. Si une erreur survient pendant la sauvegarde
2. Le système affiche le message : « Erreur : impossible d'enregistrer l'échéance. »
3. Le scénario s'arrête

Conflit de calendrier :

1. Si une autre échéance existe déjà à la même date et heure
2. Le système affiche un message : « Une échéance est déjà planifiée à cette heure. Voulez-vous remplacer? »
3. L'utilisateur choisit de confirmer ou d'annuler
 - a. S'il confirme, le scénario reprend à l'étape 9
 - b. S'il annule, le scénario s'arrête

Voici ensemble des fonctions système utilisées par les deux cas d'utilisation :

- supabase.auth.signIn() – Crée un compte utilisateur avec courriel et mot de passe
- upsert() – Ajoute ou met à jour les informations de l'utilisateur dans la table “users”

- listDeadlines() – Récupère la liste des échéances depuis la base de données
- createDeadline() – Crée une nouvelle échéance dans la base
- updateDeadline() - Met à jour une échéance existante
- deleteDeadline() - Supprime une échéance
- useState() - Gestion des états (form, items, toast, loading)
- useEffect() - Permet d'exécuter du code en réponse à des changements de dépendances ou lors du montage/démontage d'un composant
- useMemo() - Mémoire d'une valeur calculée pour éviter un recalcul inutile à chaque rendu. Utile pour optimiser les performances
- useRef() - Mémoire d'une valeur calculée pour éviter un recalcul inutile à chaque rendu. Utile pour optimiser les performances
- IonContent.scrollToBottom() - Fait défiler automatiquement le contenu d'une page vers le bas, souvent utilisé pour les formulaires ou messages dynamiques
- IonAlert - Affiche une boîte de dialogue modale pour informer l'utilisateur ou demander une confirmation
- IonToast - Affiche un message temporaire en bas de l'écran pour informer l'utilisateur d'un événement
- IonModal - Permet d'afficher un contenu modal superposé sur la page, souvent utilisé pour des sélecteurs ou formulaires secondaires
- IonDatetimeButton - Bouton qui ouvre un IonDatetime modal pour sélectionner une date et/ou heure
- Date() - Crée un objet date/heure représentant la date et l'heure actuelles ou à partir d'une chaîne ISO
- toLocaleString() - Formate un objet Date en chaîne lisible selon la localisation de l'utilisateur (date et heure)
- useHistory() - Fournit un objet history permettant de naviguer programmatiquement entre les pages (ex. : history.replace("/calendrier") pour rediriger)

Signature	Promise<Deadline> updateDeadline(id: string, payload: Partial<Deadline>)
Description	Met à jour une échéance existante dans la base de données avec les nouvelles informations fournies
Références	<p>Cas d'utilisation : Modification d'une échéance dans la page "Échéances" (DeadlinesPage).</p> <p>Étape : Lorsqu'un utilisateur édite une échéance et valide le formulaire</p>
Entrées	<ul style="list-style-type: none"> • id Type : string Signification : Identifiant unique de l'échéance à mettre à jour • payload Type : Partial<Deadline> Signification : Donnée à mettre à jour pour l'échéance
Sorties	<ul style="list-style-type: none"> • Deadline Type : Deadline Signification : L'échéance mise à jour telle qu'enregistrée dans la base
Exceptions	<ol style="list-style-type: none"> 1. Erreur si l'identifiant n'existe pas dans la base de données. 2. Erreur si la mise à jour échoue pour des raisons de connexion ou contraintes de la base.

Pré-conditions	L' id doit correspondre à une échéance existante. De plus, payload doit contenir au moins un champ à modifier et respecter les types attendus.
Post-conditions	<ul style="list-style-type: none"> L'échéance correspondante dans la base de données est mise à jour avec les nouvelles valeurs
Notes	Utilisé avec Supabase ou une API similaire Retourne un objet complet mis à jour pour synchronisation avec l'UI.

Signature	void history.replace(path: string)
Description	Redirige l'utilisateur vers une nouvelle page sans empiler l'ancienne dans l'historique de navigation (remplace la page actuelle)
Références	Cas d'utilisation : après création d'un compte, l'utilisateur est redirigé vers la page Calendrier Étape : Après inscription et mise à jour de la table users
Entrées	<ul style="list-style-type: none"> path Type : string Signification : Chemin de la route vers laquelle l'utilisateur doit être redirigé
Sorties	<ul style="list-style-type: none"> aucun retour
Exceptions	Aucune directement, mais un path peut provoquer une page 404 dans l'application
Pré-conditions	history doit être initialisé via userHistory de ReactRouter. De plus. La route fournie doit exister dans le système de routage.
Post-conditions	<ul style="list-style-type: none"> L'utilisateur voit la page /calendrier La page précédente n'est pas empilé dans l'historique du navigateur
Notes	Utilisé pour des redirections après actions critiques Diffère de history.push() qui ajoute une entrée dans l'historique

3.2 Modèle de classes

Le but de cette section est de représenter, par les modèles appropriés, « tout » ce que vous avez développé durant le sprint. Le « tout » comprend les artefacts suivants:

- 1) Une (des) base(s) de données
- 2) Des classes, dans le langage que vous avez utilisé,
- 3) Des « modules », dont la forme dépend du langage et de la technologie/framework utilisés.

Pour chaque « artefact », vous devez fournir :

- 1) Un modèle graphique
- 2) Une courte description textuelle pour expliquer/donner plus d'informations sur le modèle graphique.

Durant un sprint, vous allez peut-être développer une BD, une ou deux classes, et deux services. Il faudra documenter les trois, chacun.e selon son format. **Cependant**, ça doit être assez concis. Si les noms des classes/tables, attributs/colonnes, et méthodes/fonctions sont bien choisis, vous n'aurez pas autant

d'explications à fournir

3.2.1 Comment documenter les BDs

Pour la (les) bases de données :

- 1) Vous pouvez utiliser un modèle relationnel—si vous avez utilisé une base de données relationnelles et que vous avez fait le cours de bases de données. Sinon, un modèle de classes UML ferait l'affaire.
- 2) Pour la description textuelle : inclure un « dictionnaire de données », qui comprend les informations suivantes :

Les éléments pour décrire une table/classe, ou une classe d'association :

Nom	Utilisateur
Description	Les données fournies par chaque utilisateur lors de son inscription
Attributs	
	<uuid><id>:<identifiant de chaque utilisateur>
	<String> <fullname> : <nom de l'utilisateur>
	<String><email>:<adresse courriel de l'utilisateur>
	<String><phone>:<numéro de téléphone de l'utilisateur>
	<date><created_at>:<date de création de l'utilisateur>

Nom	Cours
Description	Le cours ajouté par l'utilisateur
Attributs	
	<uuid><id>:<identifiant de chaque cours>
	<uuid> <user_id> : <identifiant de l'utilisateur>
	<String> <code> : <sigle du cours>
	<String><title>:<titre du cours>
	<Json><passing_threshold>:<seuil de passage du cours>
	<date><created_at>:<date à laquelle le cours est ajouté par l'utilisateur>
	<date><updated_at>:<date à laquelle le cours est modifié par l'utilisateur>

Nom	Échéance
Description	L'échéance ajoutée par l'utilisateur
Attributs	
	<uuid><id>:<identifiant de chaque échéance >
	<uuid> <user_id>:<identifiant de chaque utilisateur>
	<String> <kind> : <le type de l'échéance (examen, tp, paiement, etc)>
	<String> <title> : <titre de l'échéance >
	<date><due_at>:<date et heure à laquelle l'échéance sera due>
	<String><notes>:<spécifications ou description de l'échéance>
	<date><created_at>:<date et heure à laquelle l'échéance est créée>
	<date><updated_at>:<date et heure à laquelle l'échéance est modifiée>
	<String><course_code><code d'un cours>

Notez que vous n'êtes pas obligé d'utiliser ce même format « graphique », du moment que tous les éléments

soient présents (i.e., une classe est décrite par son nom, une description textuelle, et ses attributs, et un attribut est décrit par son nom, son type, et une description textuelle).

Pour les associations en UML (qui doivent être étiquetées et leurs cardinalités indiquées dans le modèle):

Nom	<nom de l'association>		
Description	< que représente t-elle?>		
Entité source	<classe source>	Entité destination	<classe destination>

3.2.2 Comment documenter les classes dans des langages OO

Voici la présentation de la première classe qui est **Création**

Pour la documentation:

Nom	Création [extends React.FC]
Description	Composant React pour la création d'un compte utilisateur, utilisant IONIC pour l'UI et Supabase pour l'authentification
Attributs	
	String firstName : prénom saisi par l'utilisateur
	String lastName : nom saisi par l'utilisateur
	String email : email saisi par l'utilisateur
	String phone : numéro de téléphone optionnel
	String password : mot de passe saisi par l'utilisateur
	String confirm : confirmation du mot de passe
	String showAlert : contrôle l'affichage d'une alerte
	string alertMsg : message à afficher dans l'alerte
	boolean isLoading : indique si la création est en cours
	RefObject<IonContent> contentRef : référence pour le scroll automatique du contenu
Méthodes	
	Void handleCreation()
	Synopsis : valide les champs du formulaire, crée un compte via <i>Supabase</i> , affiche une alerte et redirige l'utilisateur
	Void handleFocus()
	Synopsis : scroll du IonContent jusqu'en bas avec animation pour rendre visible les champs du formulaire
	Promise<void> scrollToBottom()
	Synopsis : scroll du IonContent jusqu'en bas avec animation pour rendre visible les champs

Voici la présentation de la deuxième classe qui est **DeadlinesPage** :

Pour la documentation:

Nom	DeadlinesPage [extends React.FC]
Description	Composant React pour la gestion des échéances, permettant d'ajouter, modifier, supprimer et lister des échéances. Utilise Ionic pour l'UI et Supabase pour la persistance des données
Attributs	
	Deadline[] items : liste des échéances.
	Deadline form : formulaire actuel pour création ou modification d'échéance
	string null editingId : identifiant de l'échéance en cours de modification, sinon null
	boolean loading : indique si le chargement des données est en cours
	{ open: boolean; msg: string } toast : contrôle l'affichage d'un toast et son message
	RefObject<HTMLIonItemSliding>[] slidingRefs : références des éléments

	IonItemSliding pour contrôle de fermeture
Méthodes	
	Promise<void> fetchList()
	Synopsis : récupère la liste des échéances depuis Supabase et met à jour l'état
	void onField<K extends keyof Deadline>(key: K, val: Deadline[K])
	Synopsis : met à jour dynamiquement un champ du formulaire
	Promise<void> handleSubmit(e: React.FormEvent)
	Synopsis : ajoute ou met à jour une échéance selon si <i>editingId</i> est défini, affiche un toast de confirmation
	void resetForm()
	Synopsis : réinitialise le formulaire et quitte le mode édition.
	Promise<void> handleEdit(it: Deadline)
	Synopsis : charge une échéance dans le formulaire pour modification et ferme le <i>sliding</i>
	Promise<void> handleDelete(id?: string)
	Synopsis : supprime une échéance par son ID et ferme le <i>sliding</i> , affiche un toast

4. PROCHAIN SPRINT (1 À 2 PAGES, MAX)

4.1 Revue du sprint

4.1.1 Répartition des tâches

La répartition des tâches du **Sprint 2** s'est faite selon les modules principaux : discipline d'étude, finances et intelligence artificielle de base. Chaque membre de l'équipe a contribué à un ensemble précis de fonctionnalités, comme suit :

- **KUBARIEVA, Oryna**
 - **Issue #12 – Calcul de priorité d'événement basique (IA v1)** : conception et intégration de la formule de calcul de priorité basée sur la pondération et les jours restants avant l'échéance.
 - **Issue #13 – Barre de progression** : affichage en pourcentage de l'avancement du travail réalisé par rapport au travail restant.
- **MUNYANEZA, Ursule Joana**
 - **Issue #14 – Placement automatique de blocs d'étude (25/45/60/90 min)** : génération automatique des blocs dans les créneaux libres du calendrier.
 - **Issue #15 – Vue “Plan d'étude”** : création d'une vue quotidienne affichant la liste des tâches du jour avec un bouton *Commencer*.
- **KENFACK NANDJOU, Steve Marvyn**
 - **Module – Gestion du chronomètre d'étude** : développement complet du minuteur permettant de démarrer, mettre en pause et arrêter les sessions.
 - **Issues #16 à #19 :**
 - Allocation du temps par bloc (ex. : 25 min) ;
 - Fonction pause/reprise ;
 - Arrêt automatique du chronomètre à la fin du bloc ;
 - Gestion de la fin de session avec notification et options pour prolonger la séance (+5, +10, +15 minutes).
- **THIAM, Demba Aziz**

- **Issue #20 – Suivi de progression académique** : création de graphiques journaliers et hebdomadaires illustrant le pourcentage de tâches complétées.
 - **Issue #25 – Suivi budgétaire mensuel** : agrégation des revenus et dépenses par catégorie, avec affichage sous forme de tableau récapitulatif.
- **SOKOUDJOU SOKAMTÉ, Romuald**
 - **Module – Gestion des revenus et dépenses :**
 - **Issues #21 et #22** : ajout et modification (édition/suppression) des revenus ;
 - **Issues #23 et #24** : ajout et modification des dépenses liées aux études (coûts de trimestre, matériel, etc.).
 - **Issue #26 – Alerte de dépassement de budget** : notification automatique envoyée à l'utilisateur lorsque le budget prévu est dépassé.

4.1.2 Revue technique

Le **Sprint 1** était principalement consacré à la mise en place des fondations de l'application, notamment les modules d'**authentification**, de **gestion du profil**, du **calendrier** et des **échéances**.

Fonctionnalités complétées

- **Authentification** : interface fonctionnelle permettant à l'utilisateur de se connecter à son compte.
- **Création du profil utilisateur** : formulaire complet permettant de saisir le nom, le prénom, le courriel et le mot de passe.
- **Affichage du calendrier** : vue du calendrier opérationnelle, offrant différents modes d'affichage.
- **Personnalisation du calendrier** : possibilité pour l'utilisateur de modifier et d'adapter son calendrier selon ses préférences.
- **Ajout et modification d'échéances** : ajout d'exams, de travaux ou d'autres événements avec heure, date et informations détaillées.

Fonctionnalités partiellement complétées ou non terminées

- **Système de notifications** : encore en phase de développement au moment de la revue, à cause de problèmes d'intégration entre branches. + expliquer quelque chose de plus ????
- **Système de gestion de la seuil de passage** : amorcé, mais non finalisé durant ce sprint.

Ce qui a bien fonctionné

- Bonne collaboration entre les membres pour la mise en place du calendrier et du système d'échéances.
- L'authentification et la création de profil ont été terminées servant de base pour les sprints suivants.
- Le calendrier fonctionne correctement et son affichage est fluide, par contre il faut revoir affichage dans le version mobile

Ce qui a moins bien fonctionné

- Quelques **messages d'erreur persistants** sur la page d'authentification malgré le remplissage correct des champs.

- Petits **retards de développement** causés par la coordination entre les modules d'authentification et de calendrier.
- **Difficultés de fusion de branches**, ayant ralenti les tests d'intégration du système de notifications.

Obstacles rencontrés et solutions adoptées

- **Obstacle** : erreurs d'affichage sur la page d'authentification.
Solution : planification d'une correction dans le Sprint 2 pour améliorer la validation des champs.
- **Obstacle** : conflits lors de la fusion des branches de développement.
Solution : adoption d'une nouvelle méthode de gestion de branches et de tests avant merge.
- **Obstacle** : difficultés à tester le module de notifications sans backend complet.
Solution : simulation locale des rappels pour valider la logique avant connexion au serveur.

4.1.3 Revue gestion

Le **Sprint 1** a permis de poser les bases de l'application en développant les modules essentiels : authentification, création de profil, calendrier et gestion des échéances. La rétrospective permet d'évaluer le travail de l'équipe, les processus utilisés et les outils, afin d'identifier les points forts et les axes d'amélioration pour les prochains sprints.

Évaluation des personnes et relations

- La collaboration entre les membres a été **efficace**, avec une répartition claire des tâches selon les modules.
- La communication a été bonne malgré quelques **difficultés ponctuelles** lors des fusions de branches et les contre temps que certains membres de l'équipe ont rencontré durant la phase du développement du Sprint 1
- Les membres ont montré **de l'initiative** pour résoudre les problèmes rencontrés en demandant de l'aide aux autres et faisant les tests sur les différentes machines pour valider les points qui ont dû être à améliorer

Évaluation des processus

- La planification initiale et le suivi du sprint ont permis de **prioriser les fonctionnalités critiques**.
- L'usage du **product backlog** et des issues a facilité le suivi des tâches et la traçabilité.
- Les tests d'intégration ont été partiellement affectés par des conflits de branches, ce qui a retardé certaines validations.

Évaluation des outils

- Les outils de gestion de version et de suivi des issues (Git, GitHub/branches, Product_Backlog) ont bien supporté le développement.
- Les tests locaux ont permis de valider la plupart des fonctionnalités avant intégration, mais le **système de notifications** nécessitait encore une simulation manuelle faute de backend complet. Ainsi que la partie du seul du passage qui utilise aussi le formule pour le calculer, malgré le fait que

cette fonctionnalité a été optionnelle, il sera quand même intéressant et pertinent de l'élaborer dans le prochaine Sprint

Points positifs

- Livraison fonctionnelle des modules de base : authentification, création de profil, calendrier et ajout/modification des échéances.
- Bonne collaboration et répartition des responsabilités.
- Suivi clair des tâches et avancement visible dans le backlog.

Points à améliorer

- Résolution des **messages d'erreur persistants** sur l'authentification pour améliorer l'expérience utilisateur.
- Résolution du problème de l'affichage dans la version mobile, il y a certains éléments qui sont de bordures et cela rendre la visibilité de l'application moins pertinent et efficace
- Optimiser les **processus de fusion de branches** pour éviter les retards lors de l'intégration.
- Finaliser le **système de notifications** et la gestion de la seuil du passage.

Plan d'action pour le Sprint suivant

- Corriger les bugs identifiés sur l'authentification et tester les validations de formulaire.
- Mettre en place une procédure stricte pour la fusion des branches et les validations nécessaires qui seront fait toute de suite pour n'est pas retarder le processus du développement.
- Terminer le développement du système de notifications et le connecter au backend.
- Continuer le développement du module de gestion du seuil de passage.
- Renforcer la documentation et clarifier les responsabilités pour le Sprint 2.

4.2 Plan pour prochain sprint

Le **Sprint 2** vise à développer les fonctionnalités liées à la **discipline d'étude**, à la **gestion financière** et à une première version du **module d'intelligence artificielle (IA v1)**.

Cas d'utilisation prévus pour le Sprint 2 :

1. **Calcul de priorité des événements (IA v1)**
 - Évaluer l'importance des tâches en fonction de la pondération et des jours restants avant l'échéance.
2. **Barre de progression**
 - Afficher le pourcentage de complétion des tâches prévues pour suivre l'avancement quotidien et hebdomadaire.
3. **Placement automatique des blocs d'étude**
 - Générer des blocs d'étude de 25/45/60/90 minutes dans les créneaux libres du calendrier.
4. **Vue "Plan d'étude" quotidienne**
 - Afficher la liste des tâches du jour avec possibilité de démarrer une session d'étude via un bouton dédié.
5. **Gestion du chronomètre d'étude (timer par bloc)**

- Allocation du temps par l'utilisateur pour chaque session (ex. 25 min).
 - Pause et reprise manuelles.
 - Arrêt automatique à la fin du bloc.
 - Fin de session avec notification et option de prolongation (+5, +10, +15 minutes).
- 6. Suivi de progression académique**
- Graphiques journaliers et hebdomadaires représentant le pourcentage de tâches complétées.
- 7. Gestion des revenus et dépenses**
- Ajouter, modifier ou supprimer les revenus et les dépenses liés aux études (ex. coûts trimestriels, matériel).
- 8. Suivi budgétaire mensuel**
- Agrégation des revenus et dépenses par catégorie avec tableau récapitulatif.
- 9. Notifications financières**
- Alerte automatique en cas de dépassement du budget prévu.

5. ANNEXES

Inclure :

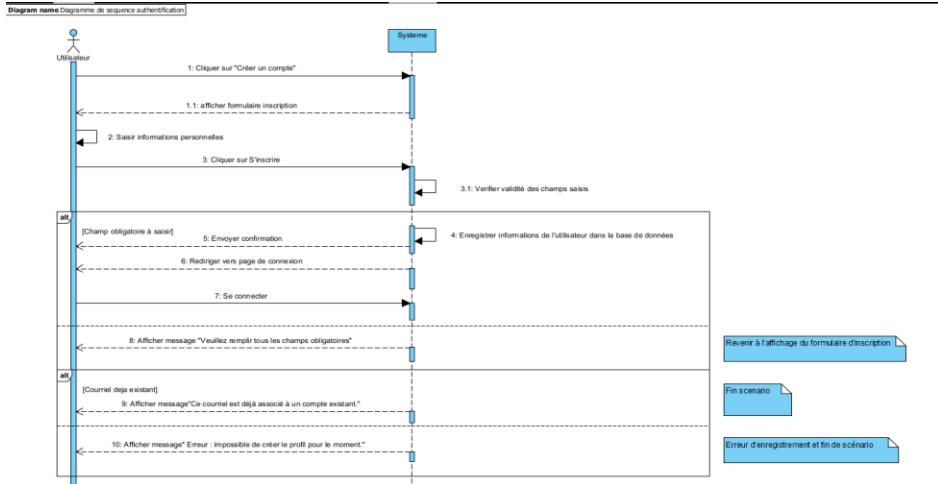
- 1) le product backlog en date d'aujourd'hui (fin du sprint courant et début du prochain sprint)
- 2) Le sprint backlog du sprint courant qui se termine
- 3) Le sprint backlog pour le prochain sprint

Les sources qui ont été utilisant durant le travail sur product backlog et les explications des fonctionnalités:

- <https://visuresolutions.com/fr/guide-d%27alm/comment-r%C3%A9diger-de-grandes-exigences/>
- [Recueillir les exigences en 6 étapes pour un projet réussi \[2025\] • Asana](#)
- [React Navigation: Router Link Redirect to Navigate to Another Page](#)
- [Open-Source UI Toolkit to Create Your Own Mobile Apps](#)
- [Supabase Docs](#)

DIAGRAMMES

- Diagramme de séquence cas Création du compte



- Diagramme de séquence Accéder à la vue Calendrier

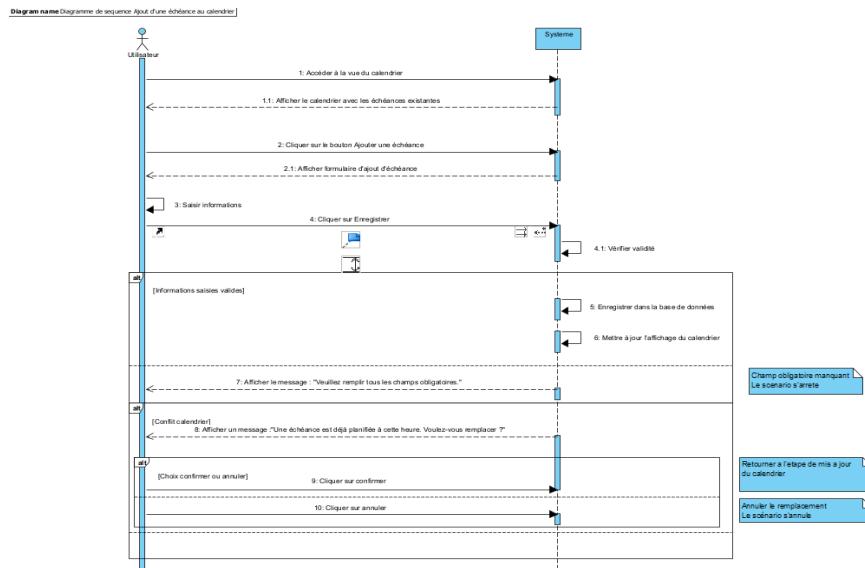
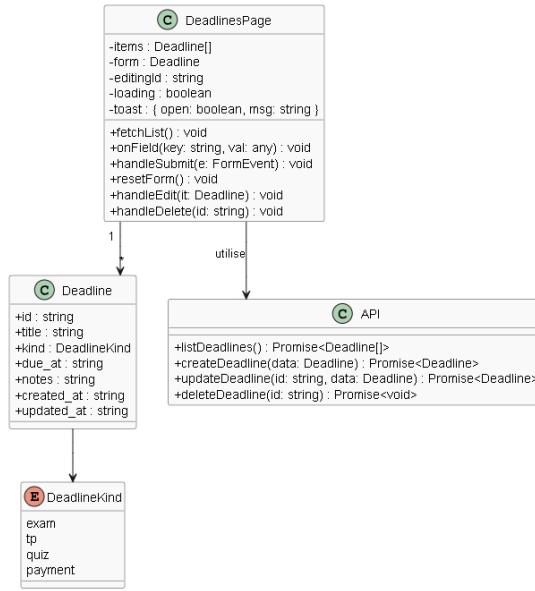


Diagramme de classe Deadline



• Diagramme de classe Creation

