Équipe 111

PolyQuiz

Plan de projet

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2024-09-14	1.0	Équipe de développement Biens livrables du projet	Damaris Calestrov
2024-09-17	1.1	Gestion de configuration Hypothèses et contraintes	Damaris Calestrov
2024-09-18	1.2	Solution proposée	Damaris Calestrov
2024-09-20	1.3	L'échéancier du projet	Lina Belloui
2024-09-20	1.4	L'entente contractuelle	Lina Belloui
2024-09-25	1.5	Introduction Gestion des risques Gestion des exigences Contrôle de la qualité	Damaris Calestrov
2024-09-2	2.0	Vérifications et corrections	Damaris Calestrov Manel Abroudj Lina Belloui Simon Cloutier Aymen Ghodbane Motassembellah Mohamed Bassiouni

Table des matières

Plan de projet

1. Introduction

Ce document présente la planification détaillée et complète du développement de la plateforme PolyQuiz. Il a pour objectif d'avoir une vision globale sur les aspects clés du projet et permettre à l'équipe de développement d'avoir une référence pour le développement et le déroulement du projet. Pour commencer, la section 2 décrit l'énoncé des travaux, dont la solution proposée qui décrit la plateforme PolyQuiz avec ses fonctionnalités principales, les hypothèses et les contraintes qui décrivent les différents éléments à prendre en compte tout au long du développement, ainsi que les biens livrables du projet présentés sous forme de tableau avec les dates de remise. La section 3 contiendra la gestion et le suivi des avancements. Plus précisément, dans cette section il sera question de la gestion des exigences qui parleront du suivi des besoins du projet, ainsi que du contrôle de la qualité qui présente les méthodes employées par l'équipe de développement pour assurer la qualité des livrables. La dernière partie de la section 3 présente les gestions de risque et configuration, donc les méthodes permettant d'organiser les versions du produit, ainsi que des artéfacts. Par la suite, l'échéancier sera disponible dans la section 4. Cette section sera présentée sous la forme d'un tableau et présente la répartition du temps pour chacune des tâches. Pour continuer, la section 5 illustre les portraits des membres de l'équipe de développement. De plus, les responsabilités de chaque membre seront nommées. Finalement, l'entente contractuelle proposée sera présentée, ainsi que le type de contrat utilisé.

2. Énoncé des travaux

2.1. Solution proposée

La proposition proposée par notre équipe est PolyQuiz, une plateforme multiplateforme et multi-mode de jeux-questionnaires. Il sera possible de rendre plateforme multiplateforme en ayant 2 clients, soit le client lourd et le client léger. Le client lourd sera implémenté pour les ordinateurs ayant comme système d'exploitation Windows 10 ou 11, tandis que le client léger aura une implémentation pour les tablettes Android. Les deux clients seront en communication grâce à un serveur, ce qui permettra aux différents joueurs de jouer ensemble sur différentes plateformes en temps réel. De plus, plusieurs parties en même temps pourront avoir lieu.

Les fonctionnalité suivantes seront disponibles sur la plateforme :

- Les joueurs pourront communiquer entre eux grâce à un clavardage pendant ou en dehors des parties à travers différents canaux.
- Chaque utilisateur aura son compte avec un pseudonyme unique, ainsi qu'un avatar. Il sera aussi possible de consulter l'historique des actions antérieures.
- En ce qui concerne les jeux, le joueur pourra choisir une partie qu'il souhaite rejoindre. Deux modes de jeu

seront disponibles, soit le mode Classique ou le mode Classique en équipe. Une évaluation des jeux pourra être faite à la fin de parties par les joueurs.

- Un joueur qui aimerait seulement observer une partie pourra le faire à partir d'une liste qui affiche les parties en cours.
- Un système de monnaie virtuelle sera disponible avec laquelle plusieurs achats pourront être faits.
- À partir de listes prédéfinies, les joueurs pourront choisir un thème et la langue de leur compte.
- Les utilisateurs pourront envoyer des demandes d'amis à partir d'une recherche. Il sera aussi possible de supprimer des amis de la liste, les bloquer ou bien lancer des parties avec des amis seulement.

Il y a des fonctionnalités qui sont propres au client lourd et d'autres propres au client léger.

- Client lourd : Il aura une vue d'administration et une vue de création. Plusieurs types de questions pourront être ajoutés au jeu et ce dernier pourra être défini public ou privé.
- Client léger : Il sera possible de changer son mot de passe. De plus, à chaque connexion, une récompense aléatoire sera attribuée aux joueurs. Finalement, un système de messagerie vocale sera disponible.

La liste des fonctionnalités qui seront implémentées se trouve dans le document Excel nommé "Liste d'exigences" remis au préalable et les différentes fonctionnalités sont plus approfondies dans le document de spécification des requis du système (SRS).

2.2. Hypothèses et contraintes

2.2.1 Ressources humaines

Le projet sera réalisé par 6 étudiants en génie logiciel. Un temps total de 1080 heures-personnes devra être accordé au développement du projet, donc chaque étudiant devra y consacrer 12h par semaine. On pose l'hypothèse que tous les membres continueront le projet jusqu'à la fin des 15 semaines et que malgré le conflit d'horaire de 3 heures pendant la séance de laboratoire de 3 membres de l'équipe, chaque personne assumera les 12 heures de travail par semaine. De plus, tous les membres de l'équipe utiliseront pour la première fois certains langages de programmation, donc une période d'apprentissage et de familiarisation avec ces langages aura lieu.

2.2.2 Équipement

On suppose que chaque membre de l'équipe de développement possède un ordinateur équipé de Windows afin de pouvoir développer le client lourd. Au besoin, les laboratoires de Polytechnique Montréal ont des ordinateurs disponibles. Toutefois, seulement une personne de l'équipe possède une tablette Android. Polytechnique Montréal a prêté à l'équipe une tablette Android (Samsung Galaxy Tab S6 Lite). L'utilisation de Android Studio sera nécessaire afin de visualiser le code écrit pour le client léger à l'aide de l'émulateur.

2.2.3 Échéancier

L’offre a été proposée à l’équipe le 26 août 2024. La réponse à l’appel d’offre devra être remise le 27 septembre 2024 et le produit final sera remis le 29 novembre 2024. On pose l’hypothèse que le projet va avoir des imprévus et qu’il va falloir s'y adapter. L'échéancier est donc sujet à changement, dépendamment de la lourdeur des tâches, des imprévus ou d’une mauvaise estimation qu’une tâche prendra. Ces changements seront déterminés suite aux rencontres hebdomadaires de l’équipe durant laquelle les coéquipiers feront part aux autres de leurs avancements, des tâches qu’ils planifient faire durant la semaine, ainsi que des difficultés rencontrées dans l’exécution des tâches assignées.

2.2.4 Technologies

On suppose que les nouvelles technologies demanderont à l’équipe un temps d’apprentissage. De plus, chaque technologie a ses caractéristiques, donc on suppose que devoir combiner les 2 clients faits chacun avec leur langage demandera beaucoup de compréhension et de temps.

2.2.5 Utilisateurs

On suppose que les utilisateurs pourront accéder à Internet de manière stable. De plus, ils auront un appareil compatible avec l’application PolyQuiz. Les utilisateurs devront aussi parler soit l’angalais ou le français et devront pouvoir lire.

2.3. Biens livrables du projet

Il y aura 5 remises durant le projet. 2 de ces remises seront la réponse à l’appel d’offre et la remise finale, tandis que les 3 autres livrables sont des rapports d’avancement qui indiquent l’avancement et l’état du projet.

Tableau 1 : Contenu des livrables et leur dates de remise

Livrable	Artefacts	Code	Date
Réponse à l’appel d’offre	- Plan du projet - Document d’architecture logicielle - Protocole de communication - Spécification des requis du	- Prototype de communication client lourd - Prototype de communication client léger - Code source et exécutable du	27 septembre 2024

	système (SRS)	produit (client lourd, client léger, serveur)	
Produit final	- Plan du projet mis à jour - Document d'architecture logicielle mis à jour - Protocole de communication mis à jour - Spécification des requis du système (SRS) mis à jour - Plan de tests logiciels - Résultats des tests logiciels	- Code source et exécutable du produit (client lourd, client léger, serveur)	29 novembre 2024
Rapport d'avancement 1	Rapport d'avancement 1	∅	3 octobre 2024
Rapport d'avancement 2	Rapport d'avancement 2	∅	31 octobre 2024
Rapport d'avancement 3	Rapport d'avancement 3	∅	5 décembre 2024

3. Gestion et suivi de l’avancement

3.1. Gestion des exigences

Les exigences à implémenter seront disponibles dans le SRS présenté au client lors de la réponse à l’appel d’offre. L’équipe de développement devra implémenter toutes les fonctionnalités essentielles et 50% des exigences souhaitables proposées. Toutes les exigences seront listées sur GitLab seront sous la responsabilité de un ou plusieurs membres. Il sera possible de suivre l’avancement dans les tâches sur GitLab, car les membres mettront à jour les la liste des exigences implémentées. De plus, des rencontres hebdomadaires seront planifiées afin de partager les avancements de chaque membre et pour intégrer les différentes parties Si les membres de l'équipe de développement décident d’implémenter une autre exigence souhaitable plutôt qu’une qui a été prévue car les priorités ont changé dans l’équipe par rapport aux fonctionnalités, les artéfacts (SRS, architecture logicielle et le protocole de communication) devront être mis à jour pour détailler les fonctionnalités de l’exigence, changer les tâches à faire sur GitLab et informer le client de ce changement. Un responsable sera désigné afin de faire les changements dans les artéfacts et sur GitLab, afin de permettre à l’équipe de ne pas se perdre dans les changements. En cas de changement d’une exigence, plusieurs méthodes seront alors utilisées afin de gérer l’impact de ces changements. Premièrement, des réunions d’équipes seront faites, afin de revoir l'échéancier et déterminer quelles sont les priorités. Ces rencontres visent à relever comment les nouveaux changements impactent l’architecture du code du projet, ainsi que la planification. De plus, une réévaluation des ressources devra être faite, afin de s’assurer que le développement des nouveaux changements pourra se faire fluidement. Pour continuer, afin de corriger les changements et les imprévus, l’équipe devra faire preuve de beaucoup de flexibilité en adoptant une approche agile qui permet de s’ajuster rapidement aux changements, sans toutefois nuire à la qualité du produit. Des scrums réguliers seront faits et un développement itératif des différentes parties du code sera fait. C’est donc en adoptant cette stratégie que l’équipe pourra faire face aux différents changements et gérer leurs impacts.

3.2. Contrôle de la qualité

En ce qui concerne les artéfacts, chaque personne sera responsable d’une partie des livrables. Avant la remise, au moins une autre personne que celle responsable de la rédaction d’une partie devra vérifier cette dernière. Si un problème de cohérence est soulevé, la personne ayant rédigé la partie de l’artéfact sera contactée pour l’informer des changements à faire et elle devra s’engager à les faire. Ensuite, le processus de vérification va recommencer jusqu’à ce que la partie rédigée soit conforme aux attentes de l’équipe.

Pour vérifier le code, celui-ci sera évalué lors de l’intégration d’une certaine partie du code. Au moins 2 personnes devront tester le code sur leur ordinateur pour s’assurer que les fonctionnalités implémentées fonctionnent bien et qu’aucune ancienne fonctionnalité n’a été brisée. De plus, ces personnes devront s’assurer que le code écrit est de bonne qualité et qu’il respecte l’architecture logicielle. Les personnes qui seront désignées pour réviser le code devront être des personnes qui n’ont pas écrit le nouveau code présent afin d’avoir une opinion objective sur le code.

Dans le cas d'une détection d'un bogue, les personnes ayant travaillé sur la partie erronée seront contactées. Une nouvelle branche sera créée à partir de la branche contenant le bogue et permettra aux développeurs de corriger le bogue avant de le faire tester de nouveau par les membres de l'équipe. Ce processus se répète jusqu'à ce que l'équipe ne trouve aucun bogue dans le code. Par la suite, le code pourra être intégré avec le reste du projet. Toutefois, même après l'intégration, il faudra tester le projet au complet pour s'assurer qu'aucun bogue n'est présent. Si c'est le cas, un nouveau processus comme mentionné précédemment sera relancé. Afin de ne pas perdre de vue les tâches de chaque membre, à chaque fois qu'un bogue devra être réglé, une tâche sur GitLab sera rajoutée avec le nom du bogue et sera assignée à la personne responsable de régler le bogue. Ce même processus devra être exécuté si en testant les fonctionnalités, un membre de l'équipe se rend compte d'un bogue. Il devra donc contacter la personne ayant écrit le code et les étapes s'enchaînent comme mentionné précédemment.

De plus, le plan de tests élaboré décrit plus en détail comment le code sera testé et les stratégies de vérification.

3.3. Gestion de risque

<1> - Nouvelles technologies				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
7	Tous les membres de l'équipe de développement utiliseront pour la première fois Kotlin et Electron. Ceci ralentira l'implémentation des nouvelles fonctionnalités, vu que chaque membre devra apprendre le fonctionnement des nouvelles technologies.	E	- Temps d'implémentation d'une fonctionnalité - Nombre de bogues rencontrés	L'équipe devra accorder plus de temps aux recherches pour se familiariser avec les technologies. Si un membre trouve une ressource intéressante, il va la partager avec le reste de l'équipe. Pour les fonctionnalités en début de projet, le temps estimé pour la réalisation d'une tâche devra inclure le temps d'apprentissage de la technologie.

<2> - Architecture logicielle

Ampleur	Description	Impact	Facteurs	Stratégie de gestion
4	Si l'architecture logicielle n'est pas de bonne qualité, l'implémentation du côté client lourd et client léger sera très compliquée et entrainera des bogues.	E	- Nombre de bogues rencontrés - Nombre de responsabilité des services	Il sera nécessaire de bien réviser le document d'architecture pour s'assurer que l'architecture logicielle est très bonne. Si en implémentant il y a des erreurs remarquées, il faudra mettre à jour l'architecture pour optimiser le code.

<3> - Respect de l'échéancier				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Si l'échéancier n'est pas respecté, il y a un énorme risque de ne pas livrer toutes les exigences promises au client. Si ceci se produit, le client n'aura pas la satisfaction et l'équipe de développement sera pénalisée.	C	- Nombre de fonctionnalités implémentées	À chaque rencontre hebdomadaire, chaque membre devra expliciter les tâches faites, les tâches qu'il fera la semaine suivante, ainsi que les problèmes qu'il rencontre. De cette façon, il sera facile de faire un suivi de l'avancement des tâches et de rediviser les tâches pour équilibrer la division des tâches.

<4> - Disponibilité des membres				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**

5	Ayant des horaires très différents, les membres de l'équipe ne seront pas toujours disponibles au même moment, ce qui peut nuire au développement du projet. De plus, plusieurs membres de l'équipe travaillent et ils sont tous à temps plein, ce qui les oblige à accorder moins de temps au projet.	C	- Nombre de fonctionnalités implémentées - Nombre de bogues non réglés - Temps accordé au développement des fonctionnalités	Il faudra faire des échéanciers réalistes et avoir une excellente communication entre les membres afin de planifier les rencontres et planifier le temps pour le développement de la plateforme.

<5> - Incohérence entre le client lourd et le client léger				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
4	En devant implémenter les fonctionnalités pour 2 interfaces en 2 langages de programmation différents, il se peut que les interfaces diffèrent beaucoup et que les fonctionnalités ne soient pas tout à fait exactement pareilles.	E	- Intuitivité de l'interface	Les développeurs qui travaillent sur une même fonctionnalité dans le client lourd et le client léger devront bien communiquer entre eux pour pouvoir produire la même chose dans les deux clients.

<6> - Intégration du code				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
7	Vu que les tâches seront divisées entre chaque membre de l'équipe, le code devra être intégré par la suite. Toutefois, il se peut que plusieurs développeurs aient modifié les mêmes composantes et/ou que les différentes logiques ne concordent pas, ce qui	M	- Temps accordé au développement des fonctionnalités	Il faudra intégrer le code des différents membres régulièrement, c'est-à-dire chaque fois qu'une partie du code est fonctionnelle pour éviter l'accumulation du code non intégré.

			- Nombre de bogues rencontrés	

<7> - Complexité des tâches				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Si le temps à accorder aux tâches est mal estimé et qu'une tâche prend plus de temps que prévu, les tâches ne seront pas équilibrées et très vite un membre de l'équipe se sentira submergé par la quantité de travail.	F	- Temps accordé au développement des fonctionnalités - Nombre de bogues rencontrés	Lors des rencontres hebdomadaires, il sera nécessaire de faire le point sur l'avancement et de préciser le temps passé sur une certaine tâche. Si la tâche demande plus de temps que prévu, une ou plusieurs personnes seront ajoutées à la réalisation de la tâche.

<8> - Mauvaise communication entre les membres				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	Durant le projet, il se peut qu'une mauvaise communication aie lieu entre les membres. Des informations pourront donc être omises et les membres ne seraient pas au courant de l'état dans lequel se trouvent les autres membres de l'équipe.	F	- Nombre de fonctionnalités implémentées - Nombre de bogues non réglés	Afin d'améliorer la communication dans l'équipe, des activités de cohésion seront faites pour briser la glace. De plus, chaque membre pourra informer les autres d'une certaine information à travers un serveur Discord. Aussi, la présence de chaque membre est requise aux scrums hebdomadaires afin de communiquer les avancements et autres informations importantes.

3.4. Gestion de configuration

Toutes les fonctionnalités devront être listées dans les issues sur GitLab. Chaque fonctionnalité sera assignée à un ou plusieurs coéquipiers selon la complexité de la tâche. Le tableau des issues sera mis à jour au fur et à mesure que le projet évolue, afin de permettre à l'équipe de se situer dans les différents avancements. Afin d'organiser le code du projet, plusieurs types de branches sur GitLab seront utilisées pour classer les différentes actions. Afin d'intégrer le code des différentes parties, des Merge Requests devront être lancées par les auteurs du code. Au moins 2 personnes seront responsables de vérifier les Merge Requests. Elles devront s'assurer que le code ne comporte pas de bogues et que les fonctionnalités répondent bel et bien aux besoins du client. Plusieurs types de branches sont utilisés afin de pouvoir organiser le code et les différentes versions de ce dernier. Les différents types de branches sont les suivants :

- Chaque ajout d'une fonctionnalité aura sa branche. Toutes les branches contenant une nouvelle fonctionnalité aura le nom sous le format suivant "feature/nom_fonctionnalité".
- Les tests unitaires auront leurs branches aussi. Les branches auront les noms sous le format "test/nom_fonctionnalité" et chaque branche contiendra les tests pour une certaine fonctionnalité.
- Si un bogue est découvert dans une fonctionnalité, le problème sera réglé dans une branche ayant le format suivant "hotfix/nom_bogue".
- La branche "dev" sera la branche qui contiendra toutes les fonctionnalités sans bogues. Chaque fonctionnalité devra être intégrée lorsqu'elle sera finie et un minimum de 2 coéquipiers devront la tester avant.
- Les branches "release/appel-offre" et "release/remise-projet" pour la première remise et pour la deuxième remise respectivement, contiendront les produits finaux qui seront testés par tous les coéquipiers. Suite aux différents tests, ces branches vont être intégrées au "main".
- La branche "main" sera la branche qui sera seulement touchée afin d'intégrer les branches "release/appel-offre" et "release/remise-projet" et servira à l'équipe comme remise finale pour les remises.

Les artéfacts du projet seront nommés comme suit : "Plan de projet_Equipe111", "SRS_Equipe111", "Architecture logicielle_Equipe111" et "Protocole de communication_Equipe111". À chaque modification, les coéquipiers qui feront les changements, vont écrire les modifications, ainsi que incrémenter le numéro de la version dans l'historique de révision. Si le changement est minime, l'incrémentation se fera de 0.1, mais si la modification est majeure, le chiffre avant le point sera incrémenté (exemple : 1.4 → 2.0). Le numéro sera modifié aussi sur la page de couverture de l'artéfact. Avant de remettre les artéfacts, au moins une personne devra repasser sur chaque partie des documents. À l'aide de l'historique de versions, les parties seront relues et corrigées par une personne autre que l'auteur d'une partie concernée.

En ce qui concerne les tickets sur GitLab, chaque fonctionnalité mentionnée dans le complément pédagogique constitue un ticket. Donc les tickets sur GitLab auront la forme suivante : "Nom de la fonctionnalité - Client X". La même fonctionnalité sera explicitée pour le client lourd et le client léger, donc on verra une duplication des tickets et à côté mentionné quel client le ticket vise. Ceci permettra de faire un avancé plus précis, dans le cas où la même fonctionnalité ne serait pas développée en même temps sur le client lourd et le client léger. De plus, elles seront facilement identifiables, vu qu'elles feront allusion à une tâche directement mentionnée dans le complément

pédagogique, ainsi que dans le SRS. En ce qui concerne les tickets visant les tâches pour la gestion des bogues, ils auront la forme suivante : “Fix - Nom de la fonctionnalité”. Ainsi, tout sera organisé afin de pouvoir avoir une vision globale de l’avancement du projet. Des tickets concernant la gestion du projet ne seront pas ajoutés, car un serveur Discord sera utilisé pour notifier les membres des différentes rencontres et des dates d'échéance.

4. Échéancier du projet

Sachant que nous consacrons environ 3h par crédit chaque semaine (soit 45 heures par session), et que ce projet vaut 4 crédits, nous estimons que chaque membre de l’équipe consacre au moins 12h par semaine sur le projet. Nous avons considéré le produit final comme étant plus exigeant, donc nous avons estimé environ 15 heures par semaine par personne au lieu de 12 heures. Pour ce qui en est des exigences du produit final, les heures personnes sont séparées selon le nombre de points que vaut chaque exigence (une exigence valant plus de points est nécessairement plus exigeante et prendra plus de temps à réaliser).

Tableau 2 : Échéancier du projet

Sprint	Spécifications	Tâches/fonctionnalités	Heures	Date de début	Date de fin
Réponse à l'appel d'offres			**240**	**16-08-2024**	**27-09-2024**
Sprint 1	Rencontre hebdomadaire	Planifier le sprint 1	2	**26-08-2024**	**01-09-2024**
	Scrums	Établir la liste des exigences	10		
	Scrums	Suivi des tâches et aider les coéquipiers	10		
	Artéfact	Réalisation du SRS - exigences fonctionnelles	10		
	Artéfact	Architecture logicielle	10		
	Artéfact	Réalisation du plan de projet	5		
	Prototype - Serveur	Création de compte utilisateur	10		
	Prototype - Client lourd	Interface pour la création du compte utilisateur	5		
	Prototype - Client léger	Interface pour la création du compte utilisateur	5		
	Rencontre d'équipe	Intégration du code	2		
	Code review	Tester l'application et valider le code	2		
Sprint 2	Rencontre hebdomadaire	Planifier le sprint 2	1	**02-09-2024**	**08-09-2024**
	Scrums	Suivi des tâches et aider les coéquipiers	5		
	Artéfact	Réalisation du SRS - exigences non fonctionnelles	10		
	Artéfact	Protocole de communication	6		
	Artéfact	Réalisation du plan de projet	15		
	Prototype - Serveur	Gestion des connexions	15		
	Prototype - Serveur	Clavardage	15		

	Rencontre d’équipe	Intégration du code	3		
	Code review	Tester l’application et valider le code	2		
Sprint 3	Rencontre hebdomadaire	Planifier le sprint 3	2	**09-09-2024**	**15-09-2024**
	Scrums	Suivi des tâches et aider les coéquipiers	10		
	Artéfact	Protocole de communication	15		
	Artéfact	Architecture logicielle	10		
	Prototype - Client lourd	Clavardage	10		
	Prototype - Client léger	Clavardage	12		
	Prototype - Serveur	Gestion des connexions	10		
	Rencontre d’équipe	Intégration du code	2		
	Code review	Tester l’application et valider le code	1		
Sprint 4	Scrums	Planifier le sprint 4	1	**16-09-2024**	**22-09-2024**
	Scrums	Suivi des tâches et aider les coéquipiers	7		
	Prototype - Client léger	Gestion des connexions	20		
	Prototype - Client lourd	Gestion des connexions	20		
	Rencontre d’équipe	Intégration	2		
	Code review	Tester l’application et valider le code	12		
Sprint 5	Scrum	Planifier le sprint 5	1	**23-09-2024**	**27-09-2024**
	Scrums	Suivi des tâches et aider les coéquipiers	11		
	Rencontre d’équipe	Intégration du code	5		
	Client lourd	Clavardage	20		
	Client léger	Clavardage	20		

	Code review	Tester l'application et valider le code	10		
Sprint 6	Rencontre hebdomadaire	Planifier le sprint 2	1	**28-09-2024**	**29-09-2024**
	Artéfact	Rapport d'avancement 1	4		
Produit final			**216**	**30-09-2024**	**29-11-2024**
Sprint 7	Rencontre hebdomadaire	Planifier le sprint 7	1	**30-09-2024**	**06-10-2024**
	Scrums	Suivi des tâches et aider les coéquipiers	2		
	Client lourd	Clavardage - Intégration : Clavardage à même l'environnement	15		
		Clavardage - Intégration : Clavardage avec le mode fenêtré	5		
		Clavardage-Canaux de discussion : Un seul canal de discussion existant	3		
		Clavardage-Canaux de discussion : Plusieurs canaux de discussion existant	10		
		Clavardage - Historique	5		
	Client léger	Clavardage - Intégration : Clavardage à même l'environnemen	15		
		Clavardage - Intégration : Notification système	5		
		Clavardage-Canaux de discussion : Un seul canal de discussion existant	10		
		Clavardage-Canaux de discussion : Plusieurs canaux de discussion existant	15		
		Clavardage - Historique	5		
	Rencontre d'équipe	Intégration du code	2		
	Code review	Tester l'application et valider le code	7		

Sprint 8	Rencontre hebdomadaire	Planifier le sprint 8	2	**07-10-2024**	**13-10-2024**
	Scrums	Suivi des tâches et aider les coéquipiers	2		
	Client lourd	Mode de jeu : Classique	10		
		Mode de jeu : Classique en équipe	10		
		Vue d'administration et Vue de création : QRE et questions avec images	5		
		Vue d'administration et Vue de création : Jeux-questionnaires privés et publiques	10		
		Réception et écoute des messages vocaux	5		
	Client léger	Messages vocaux	5		
		Mode de jeu : Classique	10		
		Mode de jeu : Classique en équipe	10		
	Artéfacts	Correction du SRS	15		
	Rencontre d'équipe	Intégration du code	1		
	Code review	Tester l'application et valider le code	1		
Sprint 9	Rencontre hebdomadaire	Planifier le sprint 9	1	**14-10-2024**	**20-10-2024**
	Scrums	Suivi des tâches et aider les coéquipiers	2		
	Client lourd	Monnaie virtuelle : Gains et dépenses	10		
		Monnaie virtuelle : partie avec un prix d'entrée	10		
		Observateur : Mode Observateur pour les jeux en cours	10		
	Client léger	Monnaie virtuelle : Gains et dépenses	10		

		Monnaie virtuelle : partie avec un prix d'entrée	5		
		Observateur : Mode Observateur pour les jeux en cours	10		
	Artéfacts	Correction du plan de projet	15		
	Rencontre d'équipe	Intégration du code	1		
	Code review	Tester l'application et valider le code	1		
Sprint 10	Rencontre hebdomadaire	Planifier le sprint 10	2	**21-10-2024**	**27-10-2024**
	Scrums	Suivi des tâches et aider les coéquipiers	5		
	Client lourd	Compte utilisateur et historique : Système de comptes	10		
		Compte utilisateur et historique : Paramètre de compte et statistiques	5		
		Compte utilisateur et historique : Historique détaillé	5		
		Avatar : Liste prédefinie	5		
		Avatar : Téléverser son avatar	5		
	Client léger	Compte utilisateur et historique : Système de comptes	10		
		Compte utilisateur et historique : Paramètre de compte et statistiques	5		
		Compte utilisateur et historique : Historique détaillé	5		
		Avatar : Liste prédéfinie	10		
		Avatar : Prendre une photo pour l'avatar	5		
	Artéfacts	Correction du protocole de communication	10		
	Rencontre d'équipe	Intégration du code	1		

	Code review	Tester l'application et valider le code	2		
Sprint 11	Rencontre hebdomadaire	Planifier le sprint 11	2	**28-10-2024**	**03-11-2024**
	Scrums	Suivi des tâches et aider les coéquipiers	1		
	Client lourd	Personnalisation de l'application : Configuration du thème visuel	10		
		Personnalisation de l'application : Configuration de la langue	10		
		Personnalisation de l'application : Persistance des configurations	5		
	Client léger	Personnalisation de l'application : Configuration du thème visuel	10		
		Personnalisation de l'application : Configuration de la langue	10		
		Personnalisation de l'application : Persistance des configurations	5		
	Artéfacts	Correction du plan d'architecture	15		
	Artéfact	Rapport d'avancement 2	4		
	Rencontre d'équipe	Intégration du code	1		
	Code review	Tester l'application et valider le code	2		
Sprint 12	Rencontre hebdomadaire	Planifier le sprint 12	1	**11-11-2024**	**17-11-2024**
	Scrums	Suivi des tâches et aider les coéquipiers	3		
	Client lourd	Système d'amis : Système d'envoi de demandes d'amis	20		
	Client léger	Système d'amis : Système d'envoi de demandes d'amis	20		

	Artéfacts	Exécution de tests	10		
	Rencontre d'équipe	Intégration du code	2		
	Code review	Tester l'application et valider le code	3		
Sprint 13	Rencontre hebdomadaire	Planifier le sprint 13	1	**18-11-2024**	**24-11-2024**
	Scrums	Suivi des tâches et aider les coéquipiers	3		
	Client lourd	Système d'amis : Création des parties amis	20		
		Système d'amis : Système de blocage	15		
	Client léger	Système d'amis : Création des parties amis	15		
		Système d'amis : Système de blocage	10		
	Artéfacts	Plan de test	10		
	Rencontre d'équipe	Intégration du code	1		
	Code review	Tester l'application et valider le code	2		
Sprint 14	Rencontre hebdomadaire	Planifier le sprint 14	1	**25-11-2024**	**29-11-2024**
	Scrums	Suivi des tâches et aider les coéquipiers	3		
	Client lourd	Banque de quiz	10		
	Client léger	Banque de quiz	5		
	Artéfacts	Plan de test	15		
	Artéfacts	Exécution de tests	10		
	Rencontre d'équipe	Intégration du code	5		
	Code review	Tester l'application et valider le code	7		
	Artéfact	Rapport d'avancement 3	4		
Projet			**1080**	**26-08-2024**	**29-11-2024**

5. Équipe de développement

Damaris Calestrov

Étudiante à sa 6e session à Polytechnique Montréal.

Expertise : Damaris a une bonne connaissance du Typescript, du Angular, du C++, de Java et de Python. Elle aime tout ce qui touche l'UX/UI et le développement du frontend des applications.

Responsabilités : Damaris s'occupera du client léger, du client lourd, des tests, ainsi que de l'interface utilisateur (UX/UI). De plus, elle aura la responsabilité de faire les remises.

Manel Abroudj

Étudiante à sa 6e session à Polytechnique Montréal.

Expertise : Ayant fait un stage chez Bell à l'été 2024, Manel a pu peaufiner ses connaissances en C++. De plus, grâce aux différents projets, elle maîtrise le Typescript, le Python, le HTML et le CSS. Le développement de l'interface utilisateur n'a aucun secret pour elle.

Responsabilités : Manel s'occupera principalement du client léger, ainsi que du front end. De plus, elle se chargera de faire les tests. Elle sera aussi l'analyste fonctionnel de l'équipe.

Lina Belloui

Étudiante à sa 7e session à Polytechnique Montréal.

Expertise : Suite à un projet personnel dans le développement d'une application web, Lina maîtrise le Typescript et Javascript. Elle a aussi une bonne connaissance du C++, du Python et du développement du frontend dans les applications web.

Responsabilités : Lina sera responsable du client léger, ainsi que du frontend du client lourd. Elle sera responsable des communications à l'intérieur de l'équipe de développement.

Aymen Ghodbane

Étudiant à sa 6e session à Polytechnique Montréal.

Expertise : Grâce à son stage à la session d'hiver 2024 chez la BNC, ainsi qu'à son stage à l'été 2024 chez Desjardins, Aymen a pu coder en Typescript, Javascript, Java, C++ et Python et maîtriser ces langages. Il maîtrise également le développement backend des applications.

Responsabilités : Aymen s'occupera du serveur, du client lourd et des tests automatisés. Il sera responsable du déploiement et des mises en production.

Simon Cloutier

Étudiant à sa 7e session à Polytechnique Montréal.

Expertise : Simon a une bonne connaissance du C++, de Java, ainsi que du Typescript.

Responsabilités : Simon s'occupera du client léger, ainsi que du serveur. Il sera responsable des risques potentiels du projet.

Motassem Mohamed Bassiouni

Étudiant à sa 6e session à Polytechnique Montréal.

Expertise : Suite à son stage chez Bell à l'été 2024 et à l'été 2023, Motassem est un expert en Python et en Ruby. De plus, il maîtrise le développement des APIs, ainsi que les fichiers de configuration. Il préfère coder en Python et C++.

Responsabilités : Motassem aura des responsabilités autant dans le développement du client léger que celui du client lourd, ainsi que du côté du serveur. De plus, il aura la responsabilité de planifier les rencontres.

6. Entente contractuelle proposée

L'entente contractuelle entre l'équipe 111 et l'entreprise Poly Apps proposée dans le cadre de notre projet est de type clé en main (prix ferme) puisque l'équipe 111 devra livrer un produit final en ayant un suivi minimal à chaque semaine du promoteur afin d'optimiser la gestion de projet.

L'équipe s'engage à respecter la réalisation de toutes les exigences essentielles du projet ainsi que de 50% des exigences souhaitables, tout en demeurant conscients que les exigences souhaitables peuvent être changées en cours de développement ce qui nécessitera une négociation entre les membres. L'équipe 111, composée de cinq développeurs et d'un gestionnaire de projet, s'engage à remettre le produit final au plus tard le 29 novembre 2024 ainsi qu'à respecter les causes et modalités suivantes décrits dans le document *Appel d'offres*:

Polytechnique Montréal peut rejeter une soumission dans l'un ou l'autre des cas suivants:

1. Si la charge de travail ne respecte pas les contraintes ci-dessous:

Chaque membre de l'équipe devra respecter la charge de travail de 1080 heures-personnes soit environ 180 heures consacrées par personne pour la livraison du projet.

2. Si le soumissionnaire ou l'un des employés est reconnu coupable en vertu de l'article 8 du document *Règlements des études du baccalauréat en ingénierie pour l'année 2024-2025*.

Le prix demandé est notamment basé sur les taux de 110$/h pour les développeurs et 145$/h pour le gestionnaire de projet. Ainsi, le coût total du projet s'élève à 125 100$. L'entreprise PolyApps se donne le pouvoir de re développer le produit final dans le cas du développement d'une version commerciale. PolyApps est également responsable de la mise en marché ou de la distribution du système.