Équipe 102

Fais-moi un dessin
Plan de projet

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2020-09-10	1.0	Version initiale	Équipe 102
2020-09-28	1.1	Ajout de l'échéancier	Vincent L'Ecuyer-Simard
2020-09-30	1.2	Révision du document pour la remise de la réponse à l'appel d'offres	Simon Berhanu, Vincent L'Ecuyer-Simard, Hakim Payman
2020-09-30	1.3	Mise en forme finale du document et correction du français	Vincent L'Ecuyer-Simard
2020-11-28	2.0	Modifications à la suite de la correction du livrable 1	Vincent L'Ecuyer-Simard

Table des matières

Plan de projet

1 Introduction

Ce document présente le plan de projet pour le logiciel *Fais-moi un dessin*. Tout d'abord, la section 2, *Énoncé des travaux*, décrira brièvement la solution proposée, les hypothèses sur lesquelles repose ce plan, ses contraintes et les artefacts du projet. Ensuite, la section 3, *Gestion et suivi de l'avancement*, expliquera la gestion des exigences, le contrôle de la qualité des biens livrables du projet, la gestion des risques et la gestion de configuration. La section 4, *Échéancier du projet*, décrira les principaux lots de travail, l'effort estimé pour chacun, les dates de début et de fin des principaux lots de travail et les dates de tombée des livrables. Par la suite, la section 5, *Équipe de développement*, décrira l'expertise des membres de l'équipe et leurs responsabilités respectives. Enfin, la section 6, *Entente contractuelle proposée*, présentera l'entente contractuelle présentée en réponse à l'appel d'offres.

2 Énoncé des travaux

2.1 Solution proposée

Nous proposons une application *Fais-moi un dessin* qui est un projet évolutif basé sur *PolyDessin* et répondant aux exigences du document de vision. Notre solution est un logiciel de jeu de dessin en réseau avec un client léger sur Android avec écran tactile et un client lourd sur ordinateur avec l'utilisation du clavier et de la souris. Le principe du jeu est de tenter de faire deviner à un coéquipier des mots en les dessinant. Nous proposons l'ajout de multiples fonctionnalités, tels qu'une interface de communication entre les usagers, la personnalisation de profil utilisateur, différents modes de jeu, l'ajout de joueurs virtuels intelligents, des effets visuels et sonores et un tutoriel interactif. En plus de tout cela, nous souhaitons réaliser plusieurs autres fonctionnalités souhaitables qui pourront ajouter une plus-value considérable au jeu tels qu'un tableau de classement, un choix de thème de l'application, le choix de couleur de la zone de dessin et du clavardage vocal. Présentement, certains outils ont déjà été développés sur *PolyDessin*, tels que l'outil de crayon, les paramètres du crayon (couleur, taille et opacité de la pointe), l'outil grille, l'outil d'efface et l'option refaire et défaire. Les clients lourd et léger doivent offrir des outils et fonctionnalités similaires pour une expérience uniforme et donc les fonctionnalités conservées de *PolyDessin* devront être implémentées sur le client léger.

2.2 Hypothèses et contraintes

2.2.1 Ressources humaines

Ce plan se base sur le travail et l'effort d'une équipe constituée de 6 développeurs. Nous estimons que 1080 heures-personnes sont disponibles pour la réalisation de la solution proposée dans les délais imposés. L'équipe est composée de développeurs avec des expériences variées avec plusieurs langages de programmation et plusieurs technologies. Il y aura tout de même un temps d'adaptation nécessaire pour apprivoiser les nouvelles technologies impliquées dans ce projet, particulièrement au niveau des communications en réseaux.

2.2.2 Équipement

Le client léger sera testé sur un émulateur de tablette tactile de modèle Galaxy Tab A 2019. Le développement du client lourd est fait avec Angular et est porté sur bureau grâce à Electron. Le client léger aura les mêmes fonctionnalités que le client lourd sauf la création d'une paire mot-image. Il n'y a pas de contraintes formelles au niveau du serveur. C'est donc à nous de choisir ses différentes caractéristiques. Par contre, celui-ci doit être en mesure de supporter deux parties de quatre joueurs au minimum et doit permettre aux clients d'accomplir toutes les fonctionnalités nécessaires. Nous supposons avoir accès à tout l'équipement nécessaire pour mener à bien ce projet.

2.2.3 Échéancier

L'échéancier est basé sur le fait que l'effort disponible est constant tout au long du projet. Cependant, lors des examens ou des laboratoires de mi-session, l'effort disponible pourrait varier selon chaque développeur. De plus, il est important de considérer la contrainte du travail à distance en lien avec la situation de distanciation sociale dans la performance de chacun, puisque celle-ci peut varier selon les individus. Les risques comme l'isolement, la conciliation vie privé et vie professionnelle et la difficulté du contrôle des heures de travail doivent donc être surveillés. Bien que le télétravail devienne une pratique de plus en plus courante, ce mode de travail est encore nouveau pour nous et c'est pourquoi il est suggéré de se munir d'un plan de communication évolutif et flexible.

2.3 Biens livrables du projet

Réponse à l'appel d'offres - 2 octobre 2020 (1er livrable)

- Remise d'Artefacts : Plan de projet, SRS, listes des exigences, document d'architecture logicielle et le protocole de communication.
- Remise de code : Code source et exécutable du prototype de communication (client lourd, client léger, serveur)

Produit final - 2 décembre 2020 (2e livrable)

- Mise à jour des artefacts remis précédemment
- Plan de tests
- Résultats de tests
- Remise de code : Code source et exécutable du produit final (client lourd, client léger, serveur)

3 Gestion et suivi de l'avancement

3.1 Gestion des exigences

Le document de spécification des requis (SRS) contient les requis sous forme d'exigences fonctionnelles et non-fonctionnelles pour l'application produite par notre équipe. Ce document définit ce que notre équipe aura à développer et sera le reflet de notre travail. Chacune des exigences possède un numéro d'identification unique. Chacune de ces exigences fait partie d'un lot de travail qui doit être implémenté selon l'échéancier disponible à la section 4 du présent document. À chaque fin de sprints, qui ont une durée de 1 semaine, nous organisons une rencontre d'équipe dédiée à la planification du prochain sprint selon l'échéancier et les tâches non complétées du sprint précédent. Nous définissons ensuite les tâches du prochain sprint sur Jira afin de pouvoir suivre leur avancement.

Durant le projet, les exigences pourraient être appelées à changer pour différentes raisons. Dans le cadre actuel, il est peu probable que le changement provienne des clients, puisque nous avons reçu un document détaillé dès le début du projet. Certains changements mineurs pourraient tout de même survenir à la suite d'interrogations par rapport à certains détails ambigus. Il est possible que les exigences souhaitables implémentées changent en cours de route. Ces changements surviendront dans le cas où nous réalisons qu'une exigence que nous pensions implémenter s'avère beaucoup plus complexe que prévu ou que, au contraire, une exigence que nous ne pensions pas implémenter s'avère beaucoup plus simple à réaliser que prévu.

En cas de changement aux exigences, nous mettrons à jour le SRS et une nouvelle version de celui-ci sera générée. Si le changement n'a pas été demandé par le client, il sera essentiel de l'avertir de ce changement. L'historique des versions du SRS permettra de voir précisément quelle exigence a subi un changement. Ensuite, nous discuterons en équipe des impacts de ce changement sur le calendrier et inclurons l'implémentation de ces changements au cours d'un sprint sur Jira.

3.2 Contrôle de la qualité

Tout d'abord, afin de mettre en place un contrôle de la qualité, plusieurs mécanismes seront mis en place au sein de l'équipe. Des règles de programmations seront mises en place dans le *README* afin d'uniformiser le code et d'avoir du code de qualité. Ces règles seront décrites dans un document à la racine même du projet.

Dans le but de s'assurer que les modifications faites ne produisent pas de régression dans l'application développée, un mode de fonctionnement d'intégration en continu sera adopté. Ce mode de fonctionnement sera mis en place avec les principes de TDD (*test driven development*), dans l'optique d'avoir des tests fonctionnels pour tous les requis de l'application. Le développement piloté par les tests offre un code plus modulaire, flexible et extensible. Le développement d'une fonctionnalité commencera par définir en équipe les tests fonctionnels qui doivent être faits en lien avec la tâche à accomplir et par la suite le code pour répondre à ces tests sera produit. Les différents tests à effectuer seront progressivement ajoutés au document de plan de tests qui spécifie les exigences à tester et la manière de les tester. Cela nous permet de garantir que le produit final respecte les requis établis. De plus, nous souhaitons automatiser une série de tests unitaires pour s'assurer de maintenir un contrôle de la qualité et un suivi des requis tout au long de notre développement. Ces tests permettront de nous donner plusieurs détails de l'impact sur l'ensemble de l'application avant l'intégration d'une fonctionnalité au produit final.

Notre équipe adoptera un mode travail utilisant les *merge requests*. En plus d'informer les autres des modifications apportées à une branche, il est possible de faire une revue de l'intégralité du code pour s'assurer de la qualité du code, du respect des tests, que la documentation du code est claire et que l'architecture est conforme aux exigences mises en place dans l'équipe. La revue d'une *merge request* sera réalisée par au moins 2 personnes qui devront examiner le code et émettre des commentaires de modification si nécessaire pour améliorer la qualité du code. Cela permet d'éviter de potentielles erreurs, d'améliorer la qualité du code, de simplifier le processus de révision du code et de fournir une meilleure stabilité pour le code. Avant d'initier une *merge request*, il sera de la responsabilité de l'instigateur de s'assurer que les tests unitaires réussissent et que les tests fonctionnels précédemment réussis le soient toujours.

Dans le cas où un des membres réviseurs soulève un problème dans le cadre d'une *merge request*, ce dernier devra détailler ses commentaires dans l'outil de revue intégré à Gitlab. Le membre ayant soumis la demande d'intégration devra ensuite apporter les correctifs nécessaires et demander une nouvelle révision une fois ceux-ci appliqués. Ce processus devra continuer jusqu'à ce que la demande d'intégration obtienne les 2 approbations nécessaires et qu'il ne reste plus de problème actif. Cela signifie que si 2 approbations sont acquises, mais qu'un troisième réviseur soulève des problèmes, ceux-ci doivent être réglés avant l'intégration.

Dans l'éventualité de l'existence d'une anomalie dans le code ou dans les fonctionnalités offertes par l'application, une tâche sera créée sur Jira avec une description claire de ce qui provoque cette anomalie, de l'hypothèse de la source du problème et ainsi que les ressources qui sont affectées par ce problème.

Du côté des artefacts, certaines activités d'assurance qualité seront accomplies pour chacun d'entre eux. Avant un livrable, ils devront tous être soumis en entièreté à la révision d'au moins 2 membres de l'équipe à l'aide du document d'aide à la rédaction de ceux-ci, puis corrigés grammaticalement à l'aide du logiciel Antidote.

Lors de la réception de la correction d'un livrable, les corrections proposées devront être appliquées aux artefacts. Durant le projet, du temps sera imparti dans les tâches Jira afin que les documents devant être maintenus à jour (SRS, architecture logicielle et protocole de communication) le demeurent lors de changements.

3.3 Gestion de risque

La description des risques suit la convention suivante :

- Ampleur : sur une échelle de 1 à 10, 10 étant le risque le plus élevé. Cette analyse est basée sur la probabilité d'occurrence du risque, ainsi que ses impacts.
- Description : une description textuelle du risque ainsi que les problèmes attendus.
- Impact : échelle définissant la portée du risque
    - C – critique (affecte le projet en entier)

    - E – élevé (affecte les fonctionnalités principales du système)
    - M – moyen (devrait être maîtrisable en appliquant une stratégie d'atténuation adéquate)
    - F – faible (l'acceptation du risque est une stratégie envisageable)
- Facteurs : aspects (**métriques**) du système pouvant être compromis.
- Stratégie de gestion : mesures à prendre afin de gérer le risque.

1 - Erreurs dans le design de l'architecture				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Une erreur de design pour l'architecture de l'application peut causer des problèmes tout au long du développement.	E	Le couplage et la cohésion Extensibilité et maintenabilité du code	Notre document d'architecture sera créé et mis à jour par toute l'équipe et nous chercherons ensemble une solution pour atténuer les impacts de cette architecture. Dans le cas de problèmes d'architecture majeurs, une refactorisation devra être entreprise.

2 - Mauvaise communication				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Une mauvaise communication entre les développeurs peut entraîner plusieurs conséquences. Dans le contexte où il existe beaucoup de dépendances entre l'équipe responsable du serveur et celles responsables des clients, une désynchronisation ou des malentendus peuvent entraîner des retards importants et du travail inutile. De plus, puisque certaines tâches dépendent de l'avancement d'autres tâches, il est important que tous sachent l'avancement de chacune des tâches.	M	Temps de développement	Nous avons un serveur de communication sur Discord pour nos rencontres et échanger de l'information. Lorsqu'un membre fait face à un problème ou a besoin de consulter l'avis de l'équipe, il peut demander de l'aide sur nos canaux de discussion. Des rencontres obligatoires ont lieu 2 fois par semaine pour synchroniser l'information et l'avancement des différents membres. Un tableau Kanban sur Jira toujours à jour sera utilisé pour pouvoir visualiser l'avancement des tâches en tout temps.

3 - Tests incomplets				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Des tests qui ne respectent pas les requis ou qui ne couvrent pas tous les cas possibles peuvent entraîner un faux sentiment de sécurité et nécessiter beaucoup de travail pour corriger les erreurs passées inaperçues.	E	Nombre d'erreurs Fiabilité du système	Des tests fonctionnels seront effectués pour s'assurer de suivre les requis de l'application. Ces tests seront élaborés par l'équipe, diminuant les probabilités d'oublier certaines situations. De plus, une fonctionnalité sera acceptée seulement si les anciens et les nouveaux tests fonctionnels sont réussis correctement et approuvés lors de la revue de code.

4 - Expérience des développeurs				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
4	Les membres de l'équipe ne sont pas tous encore familiarisés avec les technologies qui seront utilisées dans le développement de l'application. Le manque d'expérience peut avoir un impact sur le design de l'application, la qualité du code et l'implémentation de certaines fonctionnalités.	M	Nombre d'erreurs Fiabilité du système Justesse de l'échéancier	Une période d'apprentissage des nouvelles technologies sera mise en place au début du développement pour permettre aux membres de l'équipe d'avoir les ressources nécessaires pour effectuer une tâche donnée. Chacun aura la responsabilité de partager l'information apprise. Les tâches seront réparties selon les forces de chacun.

5 - Dépassement de l'échéancier				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
7	À la suite de la sous-estimation d'une ou plusieurs tâches, certaines fonctionnalités peuvent prendre plus de temps que prévu à développer.	M	Retard face à l'échéancier Temps de développement	Un travail en pair peut être envisagé pour accélérer le développement d'une fonctionnalité. Les membres sont responsables de demander de l'aide et de prévenir en avance s'ils ne peuvent pas accomplir une tâche à temps afin de pouvoir trouver des solutions. La durée de chaque tâche de l'échéancier a été légèrement surestimée, ceci permettra de tamponner les retards grâce aux gains réalisés ailleurs.

6 - Disposition des ressources				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
4	Avec la situation actuelle et étant donné que nous sommes tous des étudiants, les heures mises dans le développement de l'application peuvent varier. Il se peut que des membres soient absents à certaines rencontres. Cela peut avoir un impact dans l'avancement du projet selon notre planification.	M	Temps de développement Nombre de fonctionnalités développées	Si un membre s'absente trop souvent, une discussion d'équipe sera demandée. Chacun à la responsabilité de prévenir et de justifier un empêchement à une réunion.

7 - Perte de contact avec le serveur distant				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
4	Puisque le serveur est hébergé sur un site distant hors de notre contrôle, il est possible que celui-ci devienne inaccessible. Cependant, puisqu'il s'agit d'un site d'hébergement professionnel, cela est peu probable.	E	Disponibilité du système	Étant donné que nous avons la possibilité de mettre en marche un serveur localement, il sera facilement possible de modifier les clients pour qu'ils utilisent une copie du serveur distant localement au besoin. L'adresse de connexion au serveur sera donc modifiable à un seul endroit pour les clients, facilitant les changements. Nous suivrons de près par la suite la résolution du problème au niveau du serveur et changerons d'hébergement au besoin. Après le développement, il pourrait être intéressant de mettre en place un système de gestion des requêtes pouvant rediriger les requêtes vers un serveur secondaire dans le cas d'une panne du serveur primaire.

3.4 Gestion de configuration

Afin de faciliter la gestion de configuration, les artefacts de projet seront nommés par le numéro de version et par leur titre respectif. Le numéro de version devra débuter par 1.0. À chaque modification de type mineur ou majeur excluant la correction d'orthographe ou la reformulation de phrase, le document incrémente de version à 1.1 ensuite 1.2 et ainsi de suite. À la suite d'une remise au client, le document s'incrémente de version 1.X à 2.0 afin de marquer l'étape. Ainsi pour le premier livrable ce sera la version 1.X et pour le deuxième livrable la version 2.X.

Le mécanisme mis en place pour les problèmes et les changements se présente avec l'outil de gestion Git. Cet outil permet à l'équipe d'utiliser le principe de *merge request* pour examiner et faire une revue de code avant la soumission d'une fonctionnalité. Entre autres, des commentaires peuvent être faits directement sur Gitlab et ainsi faciliter la compréhension du problème. Également, l'outil de gestion Git nous permet d'intégrer *Gitflow* qui est utilisé conjointement avec les *merge request*. En outre, *Gitflow* est une conception de flux de travail Git qui définit un modèle de branchement strict conçu autour de la version du projet. Ainsi, nous avons plusieurs types de branches qui possèdent des rôles très spécifiques et qui définissent comment et quand elles

doivent interagir.

Notre projet contient 5 branches distinctes :

- Une branche **master** qui stocke l'historique des versions officielles mis en production.
- Une branche **hotfix** qui sera utilisée pour faire les corrections à la suite de la détection d'une anomalie.
- Une branche **release** qui sera utilisée pour marquer les étapes de développement et la gestion de version.
- Une branche **develop** qui servira de branche d'intégration pour les fonctionnalités.
- Une branche **feature** qui contiendra le développement de nouvelles fonctionnalités.

4 Échéancier du projet

Le Tableau 4.1 ci-bas présente les différents lots de travail répartis selon les sprints ainsi que les jalons importants.

Département	Lot de travail	Temps requis (heures-personnes)	Date de début	Date de fin
Sprint 1:10-09-2020 au 16-09-2020				
Gestion	Négociation des exigences	6	10-09-2020	16-09-2020
	Rédaction du plan de projet, excepté l'échéancier et l'entente contractuelle	12	10-09-2020	16-09-2020
	Rédaction du protocole de communication, excepté la description des paquets	6	10-09-2020	16-09-2020
	Rédaction du document d'architecture logicielle, excepté les différentes vues	6	10-09-2020	16-09-2020
	Discussion d'équipe sur la vue de déploiement du document d'architecture logicielle	6	10-09-2020	16-09-2020
	Rencontre de synchronisation hebdomadaire	6	10-09-2020	16-09-2020
	Rédaction des SRS	20	10-09-2020	16-09-2020
Total sprint 1: 62 heures-personnes de gestion et 0 heure-personne de développement				
Sprint 2:17-09-2020 au 23-09-2020				
Gestion	Mise en forme et correction des SRS	12	17-09-2020	20-09-2020
	Discussion d'équipe et rédaction de la description des paquets du protocole de	6	17-09-2020	23-09-2020

Département	Lot de travail	Temps requis (heures-personnes)	Date de début	Date de fin
	communication			
	Discussion d'équipe et rédaction de vue des cas d'utilisation du document d'architecture logicielle	6	17-09-2020	23-09-2020
	Discussion d'équipe et rédaction de vue logique du document d'architecture logicielle	12	17-09-2020	23-09-2020
	Discussion d'équipe et rédaction de vue des processus du document d'architecture logicielle	18	17-09-2020	23-09-2020
	Rédaction de la vue de déploiement du document d'architecture logicielle	6	17-09-2020	23-09-2020
	Rencontre de synchronisation hebdomadaire	6	17-09-2020	23-09-2020
Serveur	Implémentation de l'ébauche du prototype de communication	12	17-09-2020	23-09-2020
Client lourd	Implémentation de l'ébauche du prototype de communication	12	17-09-2020	23-09-2020
Client léger	Implémentation de l'ébauche du prototype de communication	12	17-09-2020	23-09-2020
Total sprint 2: 66 heures-personnes de gestion et 36 heures-personnes de développement				
20-09-2020	Jalon: 1re remise du document de SRS pour le cours LOG3000			
Sprint 3:24-09-2020 au 30-09-2020				
Gestion	Discussion d'équipe et rédaction de l'échéancier du plan de projet	18	24-09-2020	30-09-2020
	Rédaction de l'entente contractuelle du plan de projet	6	24-09-2020	30-09-2020
	Mise en forme et correction du plan de projet	6	24-09-2020	30-09-2020
	Mise en forme et correction du protocole de communication	6	24-09-2020	30-09-2020
	Mise en forme et correction du document d'architecture logicielle	12	24-09-2020	30-09-2020

Département	Lot de travail	Temps requis (heures-personnes)	Date de début	Date de fin
	Modification des SRS selon la correction de LOG3000	6	24-09-2020	30-09-2020
	Rencontre de synchronisation hebdomadaire	6	24-09-2020	30-09-2020
Serveur	Implémentation du prototype de communication	12	24-09-2020	30-09-2020
Client lourd	Implémentation du prototype de communication	12	24-09-2020	30-09-2020
Client léger	Implémentation du prototype de communication	12	24-09-2020	30-09-2020
Total sprint 3: 60 heures-personnes de gestion et 36 heures-personnes de développement				
02-10-2020	Jalon: Remise de la réponse à l'appel d'offres et des prototypes de communication			
Sprint 4: 01-10-2020 au 07-10-2020				
Gestion	Création de maquettes pour les interfaces principales	6	01-10-2020	07-10-2020
	Rencontre de synchronisation hebdomadaire	6	01-10-2020	07-10-2020
	Rencontre d'ajustement de calendrier	6	01-10-2020	07-10-2020
	Démo des avancements du sprint	8	01-10-2020	07-10-2020
Serveur	Intégration du prototype de clavardage et ajout des canaux de discussions	6	01-10-2020	07-10-2020
Client léger	Intégration du prototype de clavardage et ajout des canaux de discussions	6	01-10-2020	07-10-2020
	Investigation sur les implémentations possibles du crayon sur Android	12	01-10-2020	07-10-2020
	Implémentation de la navigation entre les différentes vues, sans fonctionnalités	6	01-10-2020	07-10-2020
Client lourd	Intégration du prototype de clavardage et ajout des canaux de discussions	6	01-10-2020	07-10-2020
	Intégration de *PolyDessin*	12	01-10-2020	07-10-2020

Département	Lot de travail	Temps requis (heures-personnes)	Date de début	Date de fin
	Implémentation de la navigation entre les différentes vues, sans fonctionnalités	6	01-10-2020	07-10-2020
Total sprint 4: 26 heures-personnes de gestion et 54 heures-personnes de développement				
Sprint 5:08-10-2020 au 14-10-2020				
Gestion	Rencontre de synchronisation hebdomadaire	6	08-10-2020	14-10-2020
	Rencontre d'ajustement de calendrier	6	08-10-2020	14-10-2020
	Démo des avancements du sprint	8	08-10-2020	14-10-2020
Serveur	Transmission de traits	6	08-10-2020	14-10-2020
	Consultation du profil utilisateur de base	6	08-10-2020	14-10-2020
Client léger	Implémentation de l'outil crayon	12	08-10-2020	14-10-2020
	Transmission et réception de traits	12	08-10-2020	14-10-2020
	Consultation profil utilisateur de base	6	08-10-2020	14-10-2020
Client lourd	Transmission et réception de traits	12	08-10-2020	14-10-2020
	Création mot-image manuelle I et II, excepté prévisualisation des images	12	08-10-2020	14-10-2020
	Consultation profil utilisateur de base	6	08-10-2020	14-10-2020
Total sprint 5: 20 heures-personnes de gestion et 72 heures-personnes de développement				
Sprint 6:15-10-2020 au 21-10-2020				
Gestion	Rencontre de synchronisation hebdomadaire	6	15-10-2020	21-10-2020
	Rencontre d'ajustement de calendrier	6	15-10-2020	21-10-2020
	Démo des avancements du sprint	8	15-10-2020	21-10-2020

Département	Lot de travail	Temps requis (heures-personnes)	Date de début	Date de fin
Serveur	Gestion des groupes de joueurs	6	15-10-2020	21-10-2020
	Réception de paires mot-image et transmission de paires mot-image aux joueurs virtuels	6	15-10-2020	21-10-2020
Client léger	Gestion des groupes de joueurs	6	15-10-2020	21-10-2020
	Implémentation du dessin par joueur virtuel	12	15-10-2020	21-10-2020
	Implémentation des outils manquants	12	15-10-2020	21-10-2020
Client lourd	Gestion des groupes de joueurs	6	15-10-2020	21-10-2020
	Implémentation du dessin par joueur virtuel	12	15-10-2020	21-10-2020
	Création mot-image assistée I	12	15-10-2020	21-10-2020
Total sprint 6: 20 heures-personnes de gestion et 72 heures-personnes de développement				
Sprint 7:22-10-2020 au 28-10-2020				
Gestion	Rencontre de synchronisation hebdomadaire	6	22-10-2020	28-10-2020
	Rencontre d'ajustement de calendrier	6	22-10-2020	28-10-2020
	Démo des avancements du sprint	8	22-10-2020	28-10-2020
Serveur	Gestion du mode sprint solo	12	22-10-2020	28-10-2020
	Interactions de base des joueurs virtuels dans le clavardage	6	22-10-2020	28-10-2020
Client léger	Gestion du mode sprint solo	24	22-10-2020	28-10-2020
Client lourd	Gestion du mode sprint solo	18	22-10-2020	28-10-2020
	Prévisualisation des dessins lors de la création de paires mot-image	12	22-10-2020	28-10-2020

Département	Lot de travail	Temps requis (heures-personnes)	Date de début	Date de fin
Total sprint 7: 20 heures-personnes de gestion et 72 heures-personnes de développement				
Sprint 8:29-10-2020 au 04-11-2020				
Gestion	Rencontre de synchronisation hebdomadaire	6	29-10-2020	04-11-2020
	Rencontre d'ajustement de calendrier	6	29-10-2020	04-11-2020
	Démo des avancements du sprint	8	29-10-2020	04-11-2020
Serveur	Gestion du mode sprint coop	12	29-10-2020	04-11-2020
	Gestion du mode mêlée générale	12	29-10-2020	04-11-2020
Client léger	Gestion du mode sprint coop	12	29-10-2020	04-11-2020
	Gestion du mode mêlée générale	12	29-10-2020	04-11-2020
Client lourd	Gestion du mode sprint coop	12	29-10-2020	04-11-2020
	Gestion du mode mêlée générale	12	29-10-2020	04-11-2020
Total sprint 8: 20 heures-personnes de gestion et 72 heures-personnes de développement				
Sprint 9:05-11-2020 au 11-11-2020				
Gestion	Rencontre de synchronisation hebdomadaire	6	05-11-2020	11-11-2020
	Rencontre d'ajustement de calendrier	6	05-11-2020	11-11-2020
	Démo des avancements du sprint	8	05-11-2020	11-11-2020
	Rédaction du plan de tests	6	05-11-2020	11-11-2020
Serveur	Gestion des historiques de parties	12	05-11-2020	11-11-2020
	Gestion des statistiques	6	05-11-2020	11-11-2020

Département	Lot de travail	Temps requis (heures-personnes)	Date de début	Date de fin
	Création des différentes personnalités des joueurs virtuels	12	05-11-2020	11-11-2020
Client léger	Consultation historique de partie et connexions/déconnexions	6	05-11-2020	11-11-2020
	Consultations statistiques du joueur	6	05-11-2020	11-11-2020
Client lourd	Consultation historique de partie et connexions/déconnexions	6	05-11-2020	11-11-2020
	Consultations statistiques du joueur	6	05-11-2020	11-11-2020
	Clavardage mode fenêtré et alternance	12	05-11-2020	11-11-2020
Total sprint 9: 26 heures-personnes de gestion et 66 heures-personnes de développement				
Sprint 10:12-11-2020 au 18-11-2020				
Gestion	Rencontre de synchronisation hebdomadaire	6	12-11-2020	18-11-2020
	Rencontre d'ajustement de calendrier	6	12-11-2020	18-11-2020
	Démo des avancements du sprint	8	12-11-2020	18-11-2020
Serveur	Gestion des thèmes utilisateur	6	12-11-2020	18-11-2020
	Transmission d'un choix de 3 mots pour le mode mêlée générale	6	12-11-2020	18-11-2020
Client léger	Gestion des thèmes utilisateur	6	12-11-2020	18-11-2020
	Réception d'un choix de 3 mots pour le mode mêlée générale	6	12-11-2020	18-11-2020
	Effets visuels et sonores	12	12-11-2020	18-11-2020
Client lourd	Gestion des thèmes utilisateur	6	12-11-2020	18-11-2020
	Réception d'un choix de 3 mots pour le mode mêlée générale	6	12-11-2020	18-11-2020

Département	Lot de travail	Temps requis (heures-personnes)	Date de début	Date de fin
	Effets visuels et sonores	12	12-11-2020	18-11-2020
	Création mot-image assistée II	12	12-11-2020	18-11-2020
Total sprint 10: 20 heures-personnes de gestion et 72 heures-personnes de développement				
Sprint 11:19-11-2020 au 25-11-2020				
Serveur	Interactions personnalisées de la part des joueurs virtuels	12	19-11-2020	24-11-2020
Client léger	Tutoriel	18	19-11-2020	24-11-2020
Client lourd	Tutoriel	18	19-11-2020	24-11-2020
	Création mot-image assistée III	12	19-11-2020	24-11-2020
Gestion	Démo des avancements du sprint	8	19-11-2020	25-11-2020
	Rencontre de synchronisation hebdomadaire	6	19-11-2020	25-11-2020
	Rencontre d'ajustement de calendrier	6	19-11-2020	25-11-2020
	Effectuer les tests du plan de tests	12	25-11-2020	25-11-2020
Total sprint 11: 32 heures-personnes de gestion et 60 heures-personnes de développement				
Sprint 12:26-11-2020 au 02-12-2020				
Serveur	En ordre de priorité: Correction des tests échoués ou amélioration de l'expérience utilisateur ou ajout de fonctionnalités souhaitables	12	26-11-2020	02-12-2020
Client léger	En ordre de priorité: Correction des tests échoués ou amélioration de l'expérience utilisateur ou ajout de fonctionnalités souhaitables	12	26-11-2020	02-12-2020
Client lourd	En ordre de priorité: Correction des tests échoués ou amélioration de l'expérience utilisateur ou ajout de fonctionnalités	12	26-11-2020	02-12-2020

Département	Lot de travail	Temps requis (heures-personnes)	Date de début	Date de fin
	souhaitables			
Gestion	Révision des documents de tests	12	26-11-2020	02-12-2020
	Mise à jour finale des documents remis lors de l'appel d'offres	24	26-11-2020	02-12-2020
	Rencontre de synchronisation hebdomadaire	6	26-11-2020	02-12-2020
	Démo des avancements du sprint	8	26-11-2020	02-12-2020
Total sprint 12: 50 heures-personnes de gestion et 36 heures-personnes de développement				
02-12-2020	Jalon: Livraison du logiciel et de la documentation finale			
Total sprint 1-12: 422 heures-personnes de gestion et 648 heures-personnes de développement				

Tableau 4.1. Échéancier des différents lots de travail selon les sprints et jalons importants

5 Équipe de développement

5.1 Abderrahim Ammour

- Étudiant de troisième année en génie logiciel à l'école Polytechnique Montréal.
- Connaissances avancées de C++, Typescript/JavaScript, Java et Python.
- Connaissances avancées de la programmation orientée objet.
- Connaissances de Angular.
- Connaissances intermédiaires de C#, Script Bash.
- Responsable du développement du client léger.

5.2 Simon Berhanu

- Étudiant de troisième année en génie logiciel à l'école Polytechnique Montréal.
- Connaissances avancées de Typescript/JavaScript, C#, Java et C++.
- Connaissances avancées de la programmation orientée objet.
- Connaissances avancées de Angular et React.
- Responsable du développement du client lourd.

5.3 Vincent L'Ecuyer-Simard

- Étudiant de troisième année en génie logiciel à l'école Polytechnique Montréal.
- Connaissances avancées de Typescript/JavaScript, C# et C++.
- Connaissances avancées de Angular et React.
- Connaissances avancées de .NET.
- Principal animateur lors des rencontres.
- Responsable du développement du client lourd.
- Responsable des tests et de l'assurance-qualité

5.4 Rostyslav Myovych

- Étudiant de troisième année en génie logiciel à l'école Polytechnique Montréal.
- Connaissances avancées de Python, C++, et Java.
- Connaissances avancées de la programmation orientée objet.
- Connaissances avancées de Pytorch.
- Responsable du développement du client léger.

5.5 Hakim Payman

- Étudiant de troisième année en génie logiciel à l'école Polytechnique Montréal.
- Connaissances avancées de C++, C#, Typescript/JavaScript et Java.
- Connaissances avancées de la programmation orientée objet.
- Connaissances intermédiaires en assurance qualité logicielle.
- Connaissances intermédiaires de Angular.
- Responsable du développement du client léger.
- Responsable des tests et de l'assurance-qualité

5.6 Xi Chen Shen

- Étudiant de troisième année en génie logiciel à l'école Polytechnique Montréal.
- Connaissances avancées de Typescript/JavaScript, C++ et Python.
- Connaissances avancées de la programmation orientée objet.
- Connaissances avancées de Angular.
- Responsable du développement du serveur.

6 Entente contractuelle proposée

Dans le cadre de la réponse à l'appel d'offres de Polytechnique Montréal et dans l'intérêt de l'évaluation de marché par PolyApps, l'entente contractuelle ci-dessous que nous proposons vise à mettre en place les conditions et les clauses sur lesquelles le contracteur (équipe 102) et le promoteur (Polytechnique Montréal) se sont mis d'accord pour amorcer la réalisation du logiciel *Fais-moi un dessin*.

6.1 Type de contrat proposé

Nous proposons un contrat à prix fixe pour la réalisation du logiciel *Fais-moi un dessin* puisque les exigences nécessaires au projet sont toutes clairement définies dans notre SRS, que peu de changements à ce document sont à prévoir et que le projet ne semble pas comporter de risques majeurs.

6.2 Obligations contractuelles du soumissionnaire

Nous nous engageons à réaliser toutes les exigences essentielles et à réaliser au moins la moitié des exigences souhaitables. Ces dernières sont énumérées dans notre document de spécification des requis du système, tel que mentionné à la section 3 du présent document. Nous nous soumettons aux clauses et conditions mentionnées dans la section 2 de l'appel d'offres soumis par Polytechnique Montréal. Nous nous soumettons aussi à l'éventualité que PolyApps redéveloppe entièrement le prototype dans le cadre de leur demande de proposition.

6.3 Obligations contractuelles du client

Le client (Polytechnique Montréal) se doit de respecter les droits, conditions et clauses mentionnés dans son appel d'offres et d'accomplir les débours liés à ce projet. Plus spécifiquement aux débours, il doit rémunérer en bonne et due forme l'équipe 102 selon les taux horaires spécifiés à l'annexe B de l'appel d'offres et selon la quantité de travail prévue à la section 4 du présent document.

6.4 Échéance des livrables

Nous nous engageons à réaliser et à remettre les 2 livrables du projet, c'est-à-dire, un prototype avec les

artefacts initiaux (1er livrable) et le produit complet avec les artefacts mis à jour (2e livrable) aux dates précisées dans le présent document (Section 2.3).

6.5 Rémunération

Étant donné la nature du contrat proposé, le client se doit de rémunérer son contracteur selon les taux horaires présentés dans son appel d'offres. Notamment, l'évaluation du prix total de livrable final prend donc en compte le taux horaire de développeur (100$/h) et le taux horaire de gestionnaire de projet (125$/h) en fonction du temps accordé à chaque lot de travail respectif à chaque taux. Grâce à notre échéancier du projet (voir section 4 du présent document), nous avons estimé l'effort total en heures-personnes du projet à 1070 heures-personnes où 422 heures-personnes sont consacrées à des tâches de gestion de projet et 648 heures-personnes à des tâches de développement. Nous évaluons donc le coût du livrable final à 117 550$.