Équipe 4

Poly Paint Pro
Plan de projet

Version 2.0

Historique des révisions

Date	Version	Description	Auteur
2018-09-08	1.0	La rédaction initiale du document plan de projet	Mohamed Laziz Taouali
2018-09-09	1.1	Rédaction de l'introduction ainsi que section 2.1	Jean Paul Cech
2018-09-09	1.2	Rédaction que de la section 2.2 et 2.3	Jean Paul Cech
2018-09-12	1.3	Rédaction de ma description dans la section 5	Audrey
2018-09-15	1.4	La modification de la partie 4 du document	Mohamed Laziz Taouali
2018-09-18	1.5	Première rédaction des risques	Audrey
2018-09-19	1.7	Première rédaction de la gestion des exigences	Jean Paul Cech
2018-09-19	1.8	Première rédaction du contrôle de la qualité	Jean Paul Cech
2018-09-20	1.9	Ajout de toutes les exigences fonctionnelles et la description dans la section 5	Ayman Amous
2018-09-23	1.10	Rédaction de l'échéancier du projet	Jean Paul Cech
2018-08-25	1.11	Finalisation du contrôle de la qualité	Jean Paul Cech
2018-8-25	1.12	Finalisation de l'entente contractuelle	Jean Paul Cech
2018-09-26	1.13	Révision et correction du document	Audrey
2018-09-28	1.14	Ajout d'une explication quant au calcul des heures, ajout d'informations dans la section 5 (responsabilités) et correction du calcul du coût de projet	Pascal et Audrey
2018-09-28	2.0	Dernière vérification	Sébastien

Table des matières

Plan de projet

1. Introduction

Ce document fait état du plan de projet pour le logiciel Poly Paint Pro. La section 2 aborde la solution proposée au projet, les hypothèses et contraintes reliées au projet ainsi que la liste des artefacts qui devront être créés durant le projet avec leurs dates prévues de livraison. Par la suite, la section 3 présente la gestion des exigences, de la qualité, du risque, et de la configuration. Ensuite la section 4 met en lumière l'échéancier du projet. De brèves descriptions des membres de l'équipe de développement se trouvent à la section 5. Finalement, la section 6 contient l'entente contractuelle proposée.

2. Énoncé des travaux

2.1. Solution proposée

Notre solution consiste au développement de Poly Paint Pro; un logiciel de dessin par trait. L'application sera disponible sous forme de client léger iOS roulant sur une tablette iPad (iPad mini 4) et d'un client lourd roulant sur un PC (Windows 10). Dans le cas du client lourd, l'application offrira le clavardage entre utilisateurs en mode fenêtré ou mode intégré. Plusieurs canaux de discussion peuvent être créés par les utilisateurs. Il est possible d'ajouter des formes de base (rectangle, ellipse, triangle), des formes de diagrammes (de classe et de cas d'utilisation) avec des points d'ancrage permettant de relier les formes entre elles et ajouter des formes de connexion (lignes et flèches). Ces formes peuvent être personnalisées en les redimensionnant, changeant leur couleur ou en modifiant leur bordure. Les utilisateurs ont aussi l'option d'éditer un projet simultanément avec d'autres utilisateurs ou un mode hors ligne en solo. La sauvegarde se fait automatiquement par intervalle de temps. Une galerie sera disponible sur un site web permettant de visualiser l'ensemble d'images créées par un utilisateur. Le tout sera accompagné d'un tutoriel pour former les utilisateurs. Le client léger présente des fonctionnalités presque identiques à l'exception de l'ajout d'effets visuels et sonores et du support de gestures de façon concurrentes dans l'interface utilisateur.

Les fonctionnalités sont détaillées de façon plus approfondie dans le document de spécifications des requis du système (SRS).

Ci-dessous est le tableau contenant les exigences fonctionnelles essentielles pour le client lourd et le client léger.

Tableau I : Liste des exigences fonctionnelles essentielles pour le client lourd et le client léger

Exigences fonctionnelles essentielles	
Client lourd	**Client léger**
Clavardage - Intégration	Clavardage - Intégration
Clavardage - Canaux de discussion	Clavardage - Canaux de discussion
Édition de formes	Édition de formes
Édition collaborative	Édition collaborative
Sauvegarde d'image et chargement	Interface utilisateur
Accessibilité des images	Sauvegarde d'image et chargement
Profil utilisateur et galerie	Accessibilité des images
Site web	Profil utilisateur et galerie
Tutoriel	Effets visuels et sonores
Galerie - J'aime et commentaires	Tutoriel
Photo de profil	Galerie - J'aime et commentaires
Gestion des amis	Photo de profil
	Gestion des amis

Ci-dessous est le tableau contenant les exigences fonctionnelles souhaitables pour le client lourd et le client léger.

Tableau II : liste des exigences fonctionnelles souhaitables pour le client lourd et le client léger

Exigences fonctionnelles souhaitables	
Client lourd	**Client léger**
Tutoriel	Interface utilisateur
Sauvegarde d'image et chargement	Sauvegarde d'image et chargement
Édition de formes	Profil utilisateur et galerie
Édition collaborative	Effets visuels et sonores
Galerie - J'aime et commentaires	Édition de formes
Accessibilité des images - Téléchargement des images à partir de la galerie	Édition collaborative
Accessibilité des images - Partage d'image par raccourci d'URL	Tutoriel
Meilleurs filtres pour chercher les images	Accessibilité des images - Partage d'image par raccourci d'URL
Contrôle de la police	Meilleurs filtres pour chercher les images
Historique des versions - Création et restauration	Contrôle de la police
Historique des versions - Création avancée	
Historique des versions - Historique	
Historique des versions - Comparaison	

2.2. Hypothèses et contraintes

2.2.1 Ressources humaines

L'équipe de développement est formée de 7 étudiants qui peuvent fournir un minimum de 15h de travail hebdomadaire. Ces étudiants sont à l'aise à programmer, mais pas nécessairement avec les langages de programmation Swift et C#. Une période d'adaptation sera nécessaire.

2.2.2 Équipement

Ce ne sont pas tous les membres de l'équipe qui possèdent une machine Mac et un PC. Quatre membres de l'équipe possèdent seulement un Mac et trois seulement un PC. Programmer sur les deux plateformes peut potentiellement représenter un défi. L'utilisation de machine virtuelle ou des ordinateurs dans le local de projet sera nécessaire.
Pour permettre le déploiement du projet, nous supposons que le client lourd sera déployé sur Windows 10. Pour le client léger, un iPad mini de 4e génération sera prêté à l'équipe afin de permettre le déploiement. L'application roulera sur le réseau sans-fil de l'école. On suppose qu'elle permet un débit de téléchargement minimal de 10 Mbps afin de fournir une connexion stable entre le client et le serveur.

2.2.3 Échéancier

L'équipe de travail a reçu la proposition de projet le 30 août 2018. La réponse à l'appel d'offres doit être soumise le 29 septembre. Le produit final doit être livré le 28 novembre 2018. L'équipe fera un suivi hebdomadaire chaque jeudi afin de faire un retour sur l'avancement du projet, des obstacles rencontrés et de discuter de la planification du travail.

2.3. Biens livrables du projet

- **28 septembre 2018** : Réponse à l'appel d'offres soit le plan de projet, le SRS, la liste d'exigences, le document d'architecture logicielle, le protocole de communication, ainsi que les prototypes de communication serveur-client lourd et serveur-client léger.
- **28 novembre 2018**: Mise à jour des artefacts remis lors de la remise précédente, le plan de tests, les résultats de tests, ainsi que le code source et exécutable du produit final.

3. Gestion et suivi de l'avancement

3.1. Gestion des exigences

Les requis du produit final ont premièrement été rédigés dans le document de spécification des requis (SRS). Ce document décrit en détail l'ensemble des fonctionnalités du produit final que l'équipe de développement aura à implémenter. Une rencontre hebdomadaire de l'équipe sera organisée afin de faire le suivi sur l'avancement des exigences. Le suivi de l'ensemble des exigences sera effectué dans l'outil de gestion de projet Redmine. Les exigences seront fragmentées en différentes demandes. À la suite d'une rencontre, si l'équipe décide de modifier une exigence, le changement sera effectué dans la demande Redmine respective. Ensuite, le changement causera également une modification du requis respectif trouvé dans la SRS. Finalement, l'équipe contactera le client afin de l'informer du changement proposé et pour obtenir son accord.

3.2. Contrôle de la qualité

L'équipe va adopter une mode de fonctionnement d'intégration continue. Ce mode de fonctionnement se base fortement sur les principes de TDD (test driven development), c'est-à-dire qu'on prévoit transformer les requis en une série de cas de tests. Le code de la fonctionnalité sera alors développé jusqu'au point où le code passe tous ces tests. Cela garantit que le produit final respectera tous les requis. Les différents types de tests seront décrits dans le document de plan de tests. Ce document spécifie les exigences à tester, ainsi que les différentes stratégies de tests. Nous avons également instauré un serveur d'intégration continue qui exécute automatiquement la série de cas de tests à chaque push (git) du développeur et produira un rapport des résultats. Cela permet aux développeurs de facilement reconnaitre l'impact du changement d'un nouveau push et de facilement revenir à l'arrière si un nouveau bogue s'introduit.

De plus, l'équipe adoptera un mode de travail utilisant les pull requests. Avant de propager un changement vers la branche principale du code, un pull request est ouvert afin de discuter et analyser ces changements avec le reste de l'équipe. Il faut qu'il y ait au moins une personne qui a révisé le code d'un autre pour que le pull request soit accepté dans la branche develop et au moins trois personnes pour la branche master. En plus de la visualisation, des commits peuvent être ajoutés aux pull requests pour améliorer les changements avant qu'il soit intégré dans la branche principale. Cela donne une couche additionnelle de protection contre des bogues potentiels.

Si un bogue est découvert, une demande sera créée dans Redmine sous la catégorie « anomalie » afin de répertorier les actions correctives nécessaires.

3.3. Gestion de risque

1 - Échec de livrer toutes les exigences				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	Échec de livrer toutes les exigences essentielles et la moitié des exigences souhaitables lors de la livraison du 28 novembre 2018. Si cela se produit, le client ne sera pas satisfait du produit livré car le contrat n'aura pas été respecté.	C	Nombre d'exigences réalisées	Il y aura un suivi hebdomadaire sur l'avancement des fonctionnalités lors de la rencontre d'équipe du jeudi. De cette façon, si des éléments du projet sont à risques, nous allons pouvoir le déceler tôt et rectifier l'échéancier au besoin.

2 - Non-respects des normes applicables				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
6	Les normes applicables sont énumérées dans le SRS à la section 4.8. Ces normes ont été établies pour que l'équipe de développement ait des standards par rapport à l'utilisation de Git et de la rédaction des tests. Évidemment, si ces normes ne sont pas respectées, il peut y avoir des problèmes lors du développement et de l'assurance qualité.	M	Nombre de problèmes d'intégration et de bogues	Chaque membre de l'équipe est responsable du respect des normes. Si un membre ne respecte pas une norme, l'équipe est responsable d'aviser le membre en question. De plus, nous avons décidé de travailler avec des pull requests afin de diminuer ce risque.

3 - Disponibilité des ressources				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
4	Étant donné que nous sommes tous des étudiants et que certains travaillent, il se peut que des membres de l'équipe ne puissent travailler autant d'heures que prévu. Il se peut même que des membres ne puissent pas assister à des réunions. Si c'est le cas, cela causera des problèmes dans l'avancement du projet et dans la planification.	F	Nombre d'heures de développement Nombre de fonctionnalités développées	Chaque membre est responsable d'aviser l'équipe de ses absences, retards et tout aspect qui pourrait affecter son rendement lors du projet. Si un membre s'absente trop souvent, une discussion d'équipe sera tenue à ce sujet.

4 - Temps de développement plus élevé que planifié				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
7	Puisque nous avons peu d'expérience dans la planification d'un tel projet et que nous ne pouvons pas prévoir les imprévus, nous risquons de prendre plus de temps pour développer certaines fonctionnalités. Cela causera des problèmes dans la planification des sprints et de la qualité du livrable.	M	Nombre de fonctionnalités développées Délai dans la planification	Chaque membre de l'équipe doit aviser le reste de l'équipe s'il est bloqué dans le développement d'une fonctionnalité. À chaque rencontre hebdomadaire, tout le monde doit faire part de tout élément bloquant ou qui pourrait augmenter le temps de développement.

5 - Expérience des développeurs				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Pour la majorité des membres de l'équipe, ce sera leur première expérience avec certaines des technologies utilisées dans ce projet. Donc, il se peut que notre manque d'expérience impacte plusieurs aspects du projet.	F	Nombre de bogues Nombre d'heures d'apprentissage Temps planifié pour développer	Chaque membre de l'équipe est responsable d'aviser l'équipe lorsqu'il rencontre des difficultés techniques pour obtenir de l'aide. L'équipe devra lire la documentation sur les technologies utilisées. De plus, nous allons répartir les tâches selon les forces de chacun.

6 - Erreurs dans le design de l'architecture				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
4	Des erreurs dans le design de l'architecture peuvent causer des problèmes tout au long du projet.	M	Nombre de bogues	Le document d'architecture sera révisé et compris par toute l'équipe avant de commencer à développer. Si des erreurs sont décelées, elles seront discutées en équipe et des modifications seront apportées.

7 - Mauvaise communication				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
6	Une mauvaise communication entre les membres de l'équipe et le client peut se produire. Par mauvaise communication, on fait référence à une communication manquante, incomplète ou incompréhensible. Cela peut entraîner la mauvaise compréhension de certains sujets.	M	Nombre de bogues Temps de développement	Nous avons établi un canal de communication principale qui est Slack. Ce canal servira à toute discussion hors des réunions et pour partager des informations. Toute information ambiguë sera discutée en personne lors de la réunion hebdomadaire.

8 - Développement incohérent avec l'architecture				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
2	Du développement incohérent avec l'architecture peut se produire s'il y a un retard dans le projet. À ce moment, les développeurs prennent le chemin le plus court pour terminer des fonctionnalités, et ce, en ne respectant pas ce qui a été établi dans le plan d'architecture.	E	Nombre de bogues	Étant donné que nous allons faire une réunion hebdomadaire de suivi, nous allons pouvoir identifier les problèmes qui pourraient avoir un impact sur l'architecture. Nous allons également réviser la planification pour empêcher d'accumuler du retard et de bâcler.

9 - Tests incomplets				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Tests incomplets signifient des tests qui ne couvrent pas tous les cas possibles d'erreurs. Si cela arrive, il y a un risque qu'il y ait des bogues dans le produit final.	E	Nombre de bogues	Des normes par rapport aux tests ont été établies à savoir le nombre de tests par fonctions et méthodes. De plus, aucune fonction/méthode ne peut pas être sur la branche master du git sans qu'il y ait de tests. Si ces normes ne sont pas respectées, une discussion d'équipe aura lieu.

10 - Mauvaise planification				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
7	Mauvaise planification des sprints et des exigences avant le développement et au cours des sprint. Puisqu'on ne peut pas être totalement assuré de la difficulté d'une fonctionnalité et de son temps de développement, il est presque impossible d'avoir une planification parfaite du projet.	E	Nombre d'heures planifiées pour chaque sprint	La planification des sprints sera revue à chaque fin de sprint pour déterminer s'il y a des fonctionnalités à mettre dans le backlog. Si nécessaire, la planification des sprints sera modifiée afin de tenir en compte de nouvelles réalités.

3.4. Gestion de configuration

Lorsqu'un bogue est détecté en production, une demande Redmine ayant comme parent la demande associée au bogue sera créée. La personne assignée à la demande devra alors créer une branche hotfix basée sur la branche master. Lorsque le bogue sera résolu, la branche hotfix en question sera fusionnée dans la branche master et develop.

L'outil de gestion Git sera utilisé pour faire le contrôle de version. Le dépôt aura 5 types de branches :

- une branche master qui contient une version du produit qui est toujours prête à entrer en production.
- une branche de développement
- une branche de release afin de supporter la préparation du prochain release
- des branches features représentant une fonctionnalité qui est en cours d'être complétée.
- des branches hotfix tel que mentionnées précédemment.

Les artefacts du projet seront toujours nommés par leur titre suivi de leur numéro de version. Le numéro de version débute toujours à 1.0. Chaque session de modifications incrémente le numéro de version d'une décimale soit 1.0 → 1.1 → 1.2 etc…

4. Échéancier du projet

4.1 Explication du calcul des heures:

Tout d'abord nous avons calculé le nombre d'heures de travail que nous disposions jusqu'à la remise de 28 novembre. Ceci correspond au calcul suivant:

9 semaines x 7 personnes x 12 heures/semaine/personne = 756 heures

Par la suite nous avons calculé le nombre de points total pour nos requis (liste des exigences). Ce chiffre correspond à 189 points. Ensuite, nous avons divisé le nombre d'heures par le nombre de points afin de trouver un indicateur du nombre d'heures requis pour obtenir un point.

Voici le calcul:

756 heures / 189 points = 4 heures / point

Pour être plus réaliste dans notre planification, nous avons ajouté un facteur de 30% à ce résultat. Donc, un point vaudrait 4 heures x 1.3, ce qui donne un résultat de 5.2 heures / point. Nous avons donc multiplié les points de chaque fonctionnalité par ce résultat afin de trouver le nombre d'heures requis par tâche, puis nous avons tenté de créer un échéancier avec une quantité d'heures égale pour chaque sprint, ce qui nous donne notre temps requis (heures/personne).

4.2 Échéancier du projet:

<table>
<tr><th>Sprint</th><th colspan="3">Tâches/fonctionnalités</th><th>Temps requis [heures/ personne]</th><th>Date de début</th><th>Date de fin</th></tr>
<tr><td>Réponse à l’appel d’offres</td><td colspan="3">SRS
Document d’architecture
Plan de projet
Protocole de communication
Prototypes</td><td>175</td><td>30-08-2018</td><td>28-09-2018</td></tr>
<tr><td rowspan="13">Sprint 1 [2 semaines]</td><td rowspan="7">Client lourd</td><td>Clavardage - Intégration</td><td>31.2 h</td><td rowspan="13">249.6</td><td rowspan="13">29-09-2018</td><td rowspan="13">10-10-2018</td></tr>
<tr><td>Clavardage - Canaux de discussion</td><td>10.4 h</td></tr>
<tr><td>Accessibilité des images</td><td>15.6 h</td></tr>
<tr><td>Profil utilisateur et galerie</td><td>31.2 h</td></tr>
<tr><td>Galerie - J’aime et commentaires</td><td>15.6 h</td></tr>
<tr><td>Photo de profil</td><td>10.4 h</td></tr>
<tr><td>Gestions des amis</td><td>20.8 h</td></tr>
<tr><td rowspan="6">Client léger</td><td>Clavardage - Intégration</td><td>31.2 h</td></tr>
<tr><td>Clavardage - Canaux de discussion</td><td>10.4 h</td></tr>
<tr><td>Accessibilité des images</td><td>15.6 h</td></tr>
<tr><td>Profil utilisateur et galerie</td><td>31.2 h</td></tr>
<tr><td>Gestion des amis</td><td>20.8 h</td></tr>
<tr><td>Photo de profil</td><td>10.4 h</td></tr>
</table>

Sprint 2 [2 semaines]	Client lourd	Édition de formes	104 h	244.4	11-10-2018	24-10-2018
		Sauvegarde d'image et chargement	10.4 h			
	Client léger	Édition de Formes	104 h			
		Sauvegarde d'image et chargement	104 h			
		Galerie - J'aime et commentaire	15.6 h			
	Exigences souhaitables		5 h			
Sprint 3 [2 semaines]	Client lourd	Édition collaborative	52 h	249.2	25-11-2018	07-11-2018
		Site web	57.2 h			
	Client léger	Édition collaborative	130 h			
	Exigences souhaitables		10 h			
Sprint 4 [2 semaines]	Client lourd	Tutoriel	10.4 h	193	08-11-2018	21-11-2018
	Client léger	Interface utilisateur	10.4 h			
		Effets visuels et sonores	4 h			
		Tutoriel	10.4 h			
	Exigences souhaitables		151.4 h			
Sprint 5 [1 semaine]	Exigences souhaitables		41.6 h	41.6	22-11-2018	28-11-2018
Remise du produit final	Révision Présentation Orale			100	22-11-2018	28-11-2018

5. Équipe de développement

Ayman Amous

Étudiant avec 74 crédits et ayant déjà fait une formation de 3 ans en informatique dans une autre université. Il a dernièrement fait un stage d'été en tant que développeur Full Stack. D'ailleurs, son travail durant le stage impactait directement le produit développé par l'entreprise. Durant sa formation, il a participé à plusieurs projets universitaires et non universitaires. Il possède actuellement beaucoup d'expérience en développement web dont les cadriciels Angular et symfony. Il maîtrise également plusieurs langages de programmation comme C++ et C#. Pour finir, il est toujours prêt à relever le défi grâce à son autonomie et sa prise d'initiative, il adore également travailler en équipe et cherche toujours comment s'améliorer. Puisqu'il a beaucoup d'expériences en C#, il programmera le client lourd.

Jean Paul Cech

Étudiant en 4e année à l'école polytechnique ayant trois stages en entreprises comme expérience. Son dernier stage fut effectué auprès de l'équipe de Dev Ops à la Banque Nationale. Son deuxième stage eut lieu chez CAE en coverage based testing d'une application C++. Son premier stage fut chez Bombardier en développement web d'une application Angular. Il maîtrise les langages de programmation C++, Java, python, javascript. Finalement, il s'adapte facilement à de nouveaux environnements de travail et aux technologies les plus récentes. Jean Paul n'a pas beaucoup d'expérience en langage Swift, mais il est prêt à relever le défi et apprendre la technologie afin d'implémenter le client léger.

Sébastien Chagnon

Étudiant en 3e année à Polytechnique Montréal, Sébastien est développeur Full Stack se spécialisant en sécurité des applications web. Il maîtrise les frameworks comme Angular et React, les serveurs Node.js ainsi que les bases de données MongoDB ainsi que le langage SQL. Sébastien apprend rapidement les nouvelles technologies. Même s'il n'a jamais encore travaillé sur des applications roulant directement sur le système d'exploitation, il saura rapidement s'adapter et contribuer de façon significative au projet. Puisqu'il est très expérimenté en application web, il se chargera du site web, du serveur, en plus de participer au développement du client lourd.

Audrey Labrie

Étudiante de 3e année comptant deux stages à son actif, en plus d'avoir été chargée de laboratoire dans divers cours de génie informatique et logiciel. Audrey vient de terminer un stage de huit mois à l'Ordre des ingénieurs du Québec en tant qu'analyste-programmeur. Dans cette organisation, elle a travaillé dans le développement d'un système d'authentification unifiée (sso), dans l'intégration de logiciels et dans le développement d'applications web en .NET avec C#. Elle a également effectué un stage chez Pratt & Whitney Canada en tant qu'analyste. Elle a travaillé dans le développement d'un logiciel de fabrication de *vanes* (pièce d'un moteur) et plus précisément dans l'analyse des requis et des processus d'affaires impliqués. En plus d'avoir des connaissances en développement web, elle possède de l'expérience dans le design d'interfaces utilisateurs, en gestion du changement, elle s'adapte facilement à l'utilisation de nouvelles technologies et est connue pour son sens de l'organisation. Ses expériences serviront donc à la réalisation de ce projet. Malgré son expérience en C#, Audrey veut implémenter le client léger et développer en langage Swift.

Pascal Lacasse

Étudiant en 3e année en génie logiciel sortant d'un stage en développement web chez Verdant Environnemental Technologies ou il s'est occupé de l'API, de l'interface, ainsi que de la base de données. Il possède donc beaucoup de connaissances quant au développement web côté serveur et côté d'interface utilisateur en plus d'avoir travaillé avec Amazon Web Services (EC2, S3, Lambda, RDS). Malheureusement, il n'a jamais programmé en C# et n'a pas beaucoup d'expérience sur un système d'exploitation Windows or Mac, mais il est très débrouillard et est confiant qu'il pourra apprendre rapidement les nouvelles technologies du projet. Il adore travailler en équipe, établir de nouvelles relations avec ses collaborateurs et espère pouvoir établir des contacts à long terme avec les membres de son équipe. Finalement, le client lourd l'attire beaucoup, ainsi que la base de données. Il travaillera donc sur ces éléments.

Mohamed Laziz Taouali

Un consultant programmeur analyste en TI recruté pour le gouvernement de Québec (Revenu Québec) passionné de la gestion des projets (Agile) et il suit les grands projets TI dans le cadre du virage numérique des entreprises. Il a participé dans plusieurs concours entrepreneuriaux dans la ville de Montréal dont centre d'entrepreneuriat Poly-Udem (4e dans la phase Esquisez INNOV INC RBC). Il est le fondateur de l'entreprise Agri Logiciel (Solutions informatiques pour les entreprises agricoles) et Tsunami IT (Intelligence artificielle pour le recrutement en IT). Il maîtrise l'environnement .Net dont le C# et le développement des applications mobiles surtout dans l'environnement IOS. Il est motivé et a l'habitude de travailler en équipe et aime les défis pour se challenger. Il programmera le client lourd en C#.

Sida Li

M. Li est un étudiant de 3ème année en génie logiciel à Polytechnique Montréal. Intéressé par la prise de décision basée sur les données, Eric a effectué son premier stage en 2017 chez Pratt & Whitney, où il a créé des tableaux de bord qui ont rendu les données des moteurs des clients facilement accessibles aux ingénieurs. Cela leur a permis d'offrir un service plus rapide et plus personnalisé. L'année suivante, il a effectué deux stages consécutifs chez Hivestack, une start-up de technologie marketing. Il a travaillé sur des algorithmes analysant de grands ensembles de données dans le but de diffuser des annonces plus pertinentes aux passants à proximité de panneaux d'affichage et écrans numériques dans le monde entier. Ses expériences professionnelles lui ont permis de travailler sur les logiciels à différents niveaux, dont l'expérience utilisateur, les clients webs, le serveur, les systèmes distribués, les bases de données à noeud unique et distribuées de type SQL, NoSQL, HDFS. Il a de l'expérience professionnel dans les langages de programmation suivantes: Scala, Java, Typescript, Python. Il a aussi un intérêt personnel pour le C++ et fait des projets personnels dans ce langage. Il travaillera sur le client léger (iOS), le serveur et les bases de données. Il fera aussi du DevOps (gestion d'infrastructure, intégration continue, déploiement).

6. Entente contractuelle proposée

Cette proposition d'entente du système Poly Paint Pro dans le cadre du projet 3 contient plusieurs conditions formant ainsi l'ensemble des clauses de l'entente contractuelle proposée par l'équipe 4. L'École Polytechnique de Montréal (PolyApps) et l'équipe 4 se sont mis d'accord sur les éléments suivants.

6.1 Type de contrat

Le type du contrat choisi par l'équipe 4 est un contrat fixe selon toutes les exigences mentionnées au niveau des documents de la livraison de l'appel d'offres.

6.2 Respecter les échéances intermédiaires

L'équipe 4 doit finir le développement du système Poly Paint Pro en fonction des itérations décrites dans l'échéancier du projet.

6.3 Respect l'échéancier final

L'équipe 4 a l'obligation de livrer le système Poly Paint Pro, ainsi que le plan de tests et les résultats de ceux-ci au plus tard le 28 novembre 2018 à 23h59. Sinon, le client peut réduire le paiement de l'équipe d'un montant de 10% pour chaque jour de retard .

6.4 Paiement

L'équipe 4 propose un contrat fixe pour ce projet. Le client doit la payer en fonction de ces salaires mentionnés dans le document de l'appel d'offres initial :
100$/heure pour le temps de développement
125$/heure pour gérer le projet
En prenant en compte que l'équipe est composée de sept développeurs, que la charge de travail pour une personne de l'équipe est de 180h et que le temps de gestion équivaut à 1/10 du temps de projet, le client s'engagera à débourser la somme suivante pour couvrir les frais du projet :
7 * 100$/h * 180h * 9/10 + 7 * 125$/h * 180h *1/10 = 113400$ + 22500$ = 129150$

6.5 La négociation de tout changement

Le client peut demander des changements de requis en tout temps. Par contre, si selon l'équipe de développement le changement peut engendrer un délai dans la date de livraison, le client doit accorder à l'équipe 4 le temps additionnel nécessaire pour effectuer le changement et payer les frais additionnels qui en découlent.