T6

Projet intégrateur LOG3900
Plan de projet

Version 1.8

Historique des révisions

Date	Version	Description	Auteur
2019-01-27	1.0	Première esquisse des sections 1, 2.1, 2.2, 2.3, 3.2, 3.4, 5	Amal Metsahel, Kelvin Tran
2019-01-28	1.1	Première version de la section 4	Kelvin Tran
2019-01-29	1.2	Corrections des sections 2.2, 2.3, 3.2, 3.4 et première esquisse de la section 3.3	Équipe 6
2019-02-02	1.3	Ajout d'information à la section 5	Hakim Kasimi, Freddy Sossa
2019-02-04	1.4	Révision de la section 3.3	Amal Metsahel
2019-02-05	1.5	Ajout d'information à la section 5	Kelvin Tran
2019-02-06	1.6	Révision complète de la section 4	Kelvin Tran
2019-02-07	1.7	Rédaction de l'entente contractuelle	Zouhair Chiguer
2019-02-08	1.8	Révision finale du document	Amal Metsahel

Table des matières

Plan de projet

1. Introduction

Ce document contient essentiellement l'information générale concernant le plan du projet. Effectivement, la section 2 contient une brève description de la solution proposée, les hypothèses, les contraintes et les biens livrables. La section 3 décrit les processus de gestions établis par l'équipe. L'échéancier est situé à la section 4 et l'information sur l'équipe de développement est située à la section 5. Finalement, l'entente contractuelle proposée se retrouve à la section 6.

2. Énoncé des travaux

2.1. Solution proposée

Le logiciel Poly Paint Pro est un projet d'évolution du logiciel Poly Paint. Ce dernier est un logiciel de dessin par trait, il offre à l'utilisateur un ensemble de fonctionnalités de base pour la création d'un dessin: tracer, sélectionner, effacer, couper, dupliquer, empiler et dépiler. Le projet proposé consiste à créer un nouveau logiciel de dessin basé sur *PolyPaint* et qui offre des nouvelles fonctionnalités à l'utilisateur. Le logiciel Poly Paint Pro offre la création des dessins en tant que diagrammes de classes et diagrammes de modélisation de processus. Il propose l'édition de dessins en mode collaboratif ainsi qu'une interface de communication entre les clients. Le projet propose une application exécutable sur des ordinateurs Windows (client lourd) et un autre exécutable sur des tablettes Android (client léger).

2.2. Hypothèses et contraintes

2.2.1 Les hypothèses sur lesquelles repose le projet sont les suivantes:

- La vitesse de réseau des utilisateurs est assez rapide et stable pour le transfert de données pour la communication et l'édition de dessins en temps réel.
- Les ordinateurs utilisés par les utilisateurs du client lourds roulent sous une version fonctionnelle de Windows 10.
- Les tablettes utilisées par les utilisateurs du client légers roulent sous une version fonctionnelle d'Android 7.0 ou plus récente.
- L'utilisateur possède assez de mémoire vive disponible pour faire rouler l'application.
- Les différents clients peuvent être localisés sur des réseaux différents.

2.2.2 Les contraintes imposées sont:

- une équipe de cinq développeurs travaillent sur ce projet.
- des ordinateurs qui sont dotés du système d'exploitation Windows 10 sont utilisés pour le développement et les tests du client lourd
- une tablette de type Nexus 9 qui roule sur Android est utilisée pour tester le client léger
- le projet débute le 8 janvier 2019
- le projet complet et fonctionnel qui comporte les documents livrables et les exécutables doit être livré le 8 avril 2019

2.3. Biens livrables du projet

2.3.1 [8 février 2019] Document du plan de projet

2.3.2 [8 février 2019] Document d'architecture logicielle

2.3.4 [8 février 2019] Document du protocole de communication

2.3.5 [8 février 2019] Document du SRS mis à jour

2.3.6 [8 février 2019] Code source et exécutable du client léger pour le prototype de communication

2.3.7 [8 février 2019] Code source et exécutable du client lourd pour le prototype de communication

2.3.8 [8 février 2019] Code source et exécutable du serveur pour le prototype de communication

2.3.9 [8 avril 2019] Document du plan de tests

2.3.10 [8 avril 2019] Document des résultats de tests

2.3.11 [8 avril 2019] Document du plan de projet mis à jour

2.3.12 [8 avril 2019] Document d'architecture logicielle mis à jour

2.3.13 [8 avril 2019] Document de protocole de communication mis à jour

2.3.14 [8 avril 2019] Document du SRS mis à jour

2.3.15 [8 avril 2019] Code source et exécutable du client léger final

2.3.16 [8 avril 2019] Code source et exécutable du client lourd final

2.3.17 [8 avril 2019] Code source et exécutable du serveur final

3. Gestion et suivi de l'avancement

3.1. Gestion des exigences

Une réunion hebdomadaire est appelée pour vérifier l'avancement des exigences. Avec l'outil *redmine*, chaque développeur met à jour l'avancement des tâches et tout changement apporté doit être reflété par une tâche sur *redmine* avec une version cible pour la livraison.

3.2. Contrôle de la qualité

Afin de pouvoir maintenir la qualité des biens livrables du projet, quelques méthodologies ont été mises en place. Premièrement, chaque artefact doit être lu au minimum une fois par tous les membres de l'équipe afin d'assurer l'homogénéité des visions du projet dans l'équipe. De plus, chaque section rédigée doit être vérifiée en profondeur par au moins une personne autre que l'auteur pour pouvoir maintenir un niveau de qualité adéquat.

D'autre part, l'implémentation de chaque fonctionnalité doit être vérifiée par au moins un autre développeur. Pour ce faire, chaque demande pour une fonctionnalité à implémenter doit posséder un vérificateur assigné. Ce vérificateur sera responsable de la vérification du bon fonctionnement de la fonctionnalité implémentée et de la qualité du code source associé. Si ce dernier soulève des bogues, une qualité inadéquate du code source ou n'importe quelle erreur, la demande sera retournée au développeur responsable avec un commentaire indiquant les défauts à corriger. La demande ne pourra pas fermer tant que le vérificateur n'approuve pas la solution. Si un bogue est soulevé après l'intégration à la branche principale, une nouvelle demande sera créée pour créer un correctif au problème rencontré.

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. Nbre d'heures supplémentaires , qualité des livrables, la note cible, nbre de points souhaitables
- Stratégie de gestion : mesures à prendre afin de gérer le risque.

<3.3.1> - Manque d'expertise en développement mobile				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Des difficultés rencontrées au niveau de l'implémentation du client léger à cause du manque d'expérience en développement mobile	M	Des heures supplémentaires de travail	Consacrer plus de temps à apprendre sans affecter le délai estimé pour implémenter l'application

<3.3.2> - Manque d'expertise en Langage de programmation C#				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
5	Des difficultés rencontrées au niveau de l'implémentation du client lourd à cause du manque d'expérience en langage de programmation C#	M	Des heures supplémentaires de travail	Consacrer plus de temps à apprendre le langage sans affecter le délai estimé pour implémenter l'application

<3.3.3> - Demande de changement d'une fonctionnalité				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
1	Le client demande un changement au cours du projet et cela peut affecter un ensemble d'exigences ou une fonctionnalité	E	La qualité des livrables peut se dégrader	D'abord, mettre à jour le SRS pour s'accommoder aux nouvelles exigences. Ensuite, adapter le code.

<3.3.4> - Conflit au sein de l'équipe				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
2	Un malentendu ou un désaccord dissipé au sein de l'équipe	F	La prise des décisions en équipe peut prend beaucoup de temps	Appliquer les méthodes de résolution de conflit apprises dans le cours HPR

<3.3.5> - Mauvaise estimation des délais				
Ampleur	**Description**	**Impact**	**Facteurs**	**Stratégie de gestion**
3	L'implémentation d'une fonctionnalité prend plus de temps que prévu, donc ça ne sera pas livré dans les délais	C	La note cible peut se détériorer	Mettre plus de temps sur cette fonctionnalité et la prioriser si elle présente une exigence essentielle

3.4. Gestion de configuration

La gestion de configuration pour les artéfacts est faite à l'aide de l'historique des révisions sur chaque document livrable. En effet, après la première version d'un document, chaque changement apporté à l'artefact devra être noté dans l'historique des révisions du document en question en incluant la date, la nouvelle version du document, la description de la modification et l'auteur. Pour chaque changement mineur à un artéfact, le numéro de version monte

de 0.1. Pour chaque changement majeur, le numéro de version monte au prochain entier. Tous les artéfacts sont accessibles sous un dossier Google Drive.

La gestion de version pour le code source se fait entièrement avec git. Chaque commit à une branche doit être accompagné d'un message clair et descriptif des modifications. Chaque semaine, une nouvelle étiquette git (tag) est créée sur la branche master pour pouvoir identifier les différentes versions du logiciel. Chaque nom de branche possède la forme [Préfix]-[Numéro de demande] : [Nom de branche]. Les préfixes utilisés sont les suivants :
M : Branches liées au client léger
D : Branches liées au client lourd
S : Branche liée au serveur

4. Échéancier du projet

Lot de travail	***Effort estimé (heures-personnes)***	***Date de début***	***Date de fin***
Rédaction de la réponse à l'appel d'offres ***(****8 janvier 2019 au 8 février 2019)*			
Rédaction du SRS	40	8 janvier 2019	8 février 2019
Rédaction du plan de projet	20	8 janvier 2019	8 février 2019
Rédaction du document d'architecture logicielle	40	8 janvier 2019	8 février 2019
Rédaction du document de protocole de communication	20	8 janvier 2019	8 février 2019
Développement du prototype de clavardage ***(****8 janvier 2019 au 8 février 2019)*			
Développement du prototype du client lourd	50	8 janvier 2019	8 février 2019
Développement du prototype du client léger	50	8 janvier 2019	8 février 2019
Développement du prototype du serveur	30	8 janvier 2019	8 février 2019
Développement du client lourd ***(****9 février 2019 au 8 avril 2019)*			
Conception de prototypes d'interface utilisateur	10	9 février 2019	16 février 2019
Intégration générale des modules d'interface utilisateur pour la navigation dans le logiciel	15	12 février 2019	18 février 2019
Clavardage à même l'environnement (Mode intégré)	15	12 février 2019	18 février 2019
Canaux de discussion pour le clavardage	20	19 février 2019	25 février 2019
Adaptation d'outils pour l'édition de base pour la collaboration	50	19 février 2019	4 mars 2019
Outils de rotation et de redimensionnement	20	19 février 2019	4 mars 2019
Outils pour l'édition de forme collaborative (formes avec points d'ancrage et formes de connexion)	40	5 mars 2019	11 mars 2019

Système de profils et galerie d'images publiques	30	5 mars 2019	11 mars 2019
Google login	10	12 mars 2019	18 mars 2019
Sauvegarde et chargement de canevas distant	20	12 mars 2019	18 mars 2019
Protection d'images et possibilité de changer l'état de protection	10	12 mars 2019	18 mars 2019
Personnalisation des formes de diagrammes	15	19 mars 2019	25 mars 2019
Personnalisation des formes de connexions	15	19 mars 2019	25 mars 2019
Contraintes de relation du diagramme de modélisation de processus	15	19 mars 2019	25 mars 2019
Clavardage en mode fenêtré et possibilité d'alterner entre les modes	10	26 mars 2019	1 avril 2019
Sauvegarde et chargement local	10	26 mars 2019	1 avril 2019
Tutoriel non interactif	10	26 mars 2019	1 avril 2019
Élaboration du plan de tests logiciels et rédaction du document de plan de tests logiciels	10	26 mars 2019	1 avril 2019
Exécution des tests logiciels et rédaction du document de résultats de tests	5	2 avril 2019	8 avril 2019
Corrections finales du logiciel	10	2 avril 2019	8 avril 2019
Développement du client léger ***(****9 février 2019 au 8 avril 2019)*			
Conception de prototypes d'interface utilisateur	10	9 février 2019	16 février 2019
Intégration générale des modules d'interface utilisateur pour la navigation dans le logiciel	15	12 février 2019	18 février 2019
Fenêtre de chat intégré et notifications	15	12 février 2019	18 février 2019
Fonctionnalités d'ajout de forme, d'efface de forme, de sélection et de déplacement en collaboration	50	19 février 2019	25 février 2019
Fonctionnalités de réinitialisation, de redimensionnement de canevas, de duplication, de coupage en collaboration	30	25 février 2019	4 mars 2019
Fonctionnalités d'empilage/dépilage, d'insertion de texte flottant, de lasso en collaboration	40	25 février 2019	4 mars 2019
Formes de diagrammes et formes de connexions	40	5 mars 2019	11 mars 2019
Système de profils et galerie d'images publiques	30	12 mars 2019	18 mars 2019
Sauvegarde et chargement de canevas distant	20	12 mars 2019	18 mars 2019

Protection d'images et possibilité de changer l'état de protection	10	12 mars 2019	18 mars 2019
Contraintes de relation du diagramme de modélisation de processus	10	19 mars 2019	25 mars 2019
Tutoriel non interactif	10	19 mars 2019	25 mars 2019
Effets visuels et sonores	5	19 mars 2019	25 mars 2019
Élaboration du plan de tests logiciels et rédaction du document de plan de tests logiciels	10	26 mars 2019	1 avril 2019
Exécution des tests logiciels et rédaction du document de résultats de tests	5	2 avril 2019	8 avril 2019
Corrections finales du logiciel	10	2 avril 2019	8 avril 2019

5. Équipe de développement

Amal Metsahel
- Étudiante de troisième année au baccalauréat en génie logiciel à l'école Polytechnique Montréal.
- Maîtrise du langage : C++, Java.
- Responsabilités :
    - Développement du client léger

Freddy-Godgives Sossa
- Étudiant de troisième année au baccalauréat en génie logiciel à l'école Polytechnique Montréal.
- Maîtrise du langage : C++, Java.
- 4 mois d'expérience avec Node.js et C#
- Responsabilités :
    - Développement du client léger
    - Chef de l'assurance qualité

Hakim Kasimi
- Étudiant de troisième année au baccalauréat en génie logiciel à l'école Polytechnique Montréal.
- Maîtrise du langage : C#, C++, Java.
- Responsabilités :
    - Développement du client lourd

Kelvin Tran
- Étudiant de troisième année au baccalauréat en génie logiciel à l'école Polytechnique Montréal.
- Maîtrise du langage : C++, C#, Java, Typescript
- 4 mois d'expérience avec Node.js
- Maîtrise des concepts de réseaux
- Responsabilités :
    - Développement du serveur
    - Maintenance de la base de données
    - Maintenance de la machine virtuelle hébergée sur le cloud

Zouhair Chiguer
- Étudiant de troisième année au baccalauréat en génie logiciel à l'école Polytechnique Montréal.
- Maîtrise du langage : C++, C#, Java, Typescript, NodeJS
- 4 mois d'expérience avec Node.js
- 8 mois d'expérience de développement en C++, Java et C#

- Responsabilités :
    - Développement du client lourd
    - Gestionnaire de projet

6. Entente contractuelle proposée

Pour répondre favorablement à cet appel d'offres et démontrer notre intérêt, nous proposons un contrat avec partage de profit vu que Polytechnique Montréal a un intérêt à commercialiser un produit sur la base de ce prototype par la PME Polyapps. Dans ce cas, nous proposons les clauses suivantes:

- Prix demandé : 72500$ se détaillant comme suit
    - Heure-personne de développement : 800 heures à raison de 100$/heure
    - Heure-personne de gestion de projet : 100 heures à raison de 125$/heure
    - Rabais de 20000$ et remplacé par la clause de partage de profit qui suit.
- Partage des profits : une redevance de 3% des ventes réalisées sur une base annuelle pour les quatre premières années de vie commerciale du logiciel.
- Changement aux exigences : en cas ou Polytechnique, Montréal désire introduire un changement aux exigences, les modalités suivantes s'appliquent:
    - Toute nouvelle exigence sera considérée comme souhaitable avec un pointage double à ce qui a été précédemment approuvé, et ne sera pas garantie dans le livrable.
    - Polytechnique Montréal a aussi le choix de nouvelle exigence qui sera évalué sur la base d'un mécanisme *TradeIn/TradeOut*, et remplacera une exigence existante à la condition que le développement de cette dernière ne soit pas déjà commencé comme spécifiée dans l'échéancier à la section 4.
    - Pour toute exigence annulée, son pointage devra être redistribué sur le reste des exigences essentielles.