Projet synthèse

420-C61-VM
420-C61-IN

Énoncé principal

Session hiver 2022

Table des matières

Sommaire

Introduction

Tel qu'il est stipulé dans le plan de cours, ce projet vous donne l'occasion de réaliser une application informatique et d'en assurer la mise en oeuvre complète. Vous devez implémenter une application logicielle fonctionnelle en passant par toutes les étapes de production typiques d'un tel projet : de la conception à la réalisation en passant par la planification, la documentation, l'exploration technologique et le prototypage.

Comme l'indique le titre du cours, ce projet se veut une synthèse des apprentissages que vous accumulés pendant votre formation. Il est donc essentiel de mettre à profit toutes les connaissances que vous avez acquises afin de rendre hommage à l'informaticien.ne que vous êtes rendu.e!

Vous devez savoir que la réussite de ce cours est obligatoire à l'obtention de votre diplôme d'études collégiales.

Forme du cours

Le cours de Projet synthèse se déroule à raison de quatre blocs horaires par semaine en classe et il est attendu que vous produisiez une somme de travail équivalente à l'extérieur des heures de cours. En définitive, vous devriez avoir réalisé environ 120 heures de travail sur ce projet.

Outre quelques notions présentées en début de session, le cours ne présente pas de nouvelles matières spécifiques. Par conséquent, à l'exception des éléments mentionnés, il n'y aura pas de cours magistraux et d'évaluations spécifiques sur du nouveau contenu.

Cependant, tous les projets requièrent l'approfondissement de nouvelles connaissances et de nouvelles compétences. Ce sera à vous d'identifier et d'acquérir les éléments requis au succès de votre projet. Ce sera votre responsabilité d'aller chercher les ressources requises[1] afin de vous approprier les connaissances indispensables à la réussite de ce dernier. D'ailleurs, le cours Veille technologique vous donnera l'occasion d'approfondir un sujet pertinent facilitant le développement de votre projet synthèse via l'utilisation de technologies spécifiques ou de connaissances plus poussées sur certains sujets.

Sujet du projet

À l'exception de quelques considérations présentées plus loin dans ce document, le sujet de votre projet est ouvert. Par conséquent, vous devez identifier tôt le sujet que vous voulez aborder et de quelles façons vous voulez le faire.

[1] Les ressources peuvent être de diverses natures : les enseignants du département selon leurs spécialités respectives, des documents de référence sur les outils à exploiter, des tutoriaux en ligne, des exemples de code, ...

Même si vous avez carte blanche pour le projet, il faut tout de même que le sujet soit validé avec l'enseignant. Il est obligatoire que votre sujet soit présenté et accepté par l'enseignant avant que vous le considériez valide. Cette validation doit avoir lieu au plus tard, à la deuxième semaine de la session!

Il va sans dire que votre projet doit être exempt d'éléments de morale douteuse et répondre à un niveau d'éthique élémentaire. Ceci est la seule contrainte non technologique que vous devez respecter.

Attentes liées à la réalisation du projet

Il est important de comprendre que ce projet vise à démontrer votre capacité à produire un logiciel de qualité.

Même s'il est attendu que votre logiciel soit opérationnel, il ne suffit pas que ce dernier soit fonctionnel. Il est fondamental que le résultat présente un niveau de qualité logiciel satisfaisant. Autrement dit, il ne suffit pas de réaliser une multitude de fonctionnalités mais plutôt de produire une architecture logicielle donnant de la valeur à votre projet. On misera davantage sur des critères de qualité, de structure, de modularité, de maintenabilité que sur des considérations cosmétiques ou le fait d'avoir un éventail de fonctionnalités similaires.

Gardez en tête que ce projet vise davantage à démontrer vos capacités à un professionnel de l'industrie plutôt qu'à impressionner un client ou un ami ne connaissant rien au développement informatique.

De plus, puisque vous vous apprêtez à entreprendre la prochaine étape de votre parcours professionnel, soit le marché du travail ou l'université, vous pouvez considérer ce projet comme une opportunité de présenter ce que vous êtes capable de faire. Il est attendu que ce projet soit du niveau technique auquel peut s'attendre un employeur qui considère l'embauche d'un finissant du niveau collégial.

Contraintes

Malgré le fait que le sujet de votre projet soit ouvert, il existe tout de même plusieurs contraintes qui doivent être respectées. On peut regrouper ces obligations en 4 catégories :

1. Contraintes générales
2. Évaluations
3. Contraintes technologiques
4. Contraintes de contenu technique

Chacune de ces contraintes sont décrites en détails dans les sections subséquentes. Toutefois, voici les plus importantes :

- Le projet doit être réalisé en équipe de deux étudiants (exceptionnellement, une équipe de trois étudiants sera acceptée si la classe compte un nombre impair d'étudiants). Aucun projet réalisé seul ne sera accepté.
- Vous devez respecter l'échéancier donné.
- Vous devez remettre tous les documents demandés.
- Vous devez mettre une emphase particulière sur la planification du projet et un effort remarqué sur la qualité de votre logiciel.

- À la base, vous devez utiliser des technologies que vous avez vu dans le cadre de votre DEC. Vous pouvez toutefois utiliser des technologies dérivées qui n'ont pas été présentées (une certaine librairie en JavaScript par exemple).
- Vous ne pouvez pas utiliser bêtement un « framework » pour lequel vous avez très peu de développement informatique à réaliser.
- Vous devez mentionner toutes vos sources.
- Vous devez utiliser GIT.

Stratégies de développement

Pour donner suite aux notions que vous avez acquises lors des cours précédents, le développement et la gestion du projet s'effectueront selon la méthode Scrum respectant bien le Manifeste agile.

On encourage et valorise sérieusement la pratique de la programmation en binôme. Il est aussi recommandé de mettre intelligemment en pratique les notions de conception UML conjointement avec le développement piloté par les tests.

Dans cet esprit, la réalisation du projet est découpée en quatre étapes distinctes :

1.	Identification, conception et planification	5 semaines
2.	Sprint 1 : mise en place	3 semaines
3.	Sprint 2 : avancement	3 semaines
4.	Sprint 3 : déploiement	4 semaines

Pour chaque étape, vous aurez plusieurs évaluations décrites dans les sections subséquentes de ce document.

Le projet qui vous est demandé est exigent pour le temps que vous avez de disponible. Ainsi, vous allez devoir gérer vos activités très efficacement afin d'éviter les pièges typiques de tels développements. D'ailleurs, vous devez vous y mettre dès la première semaine afin de ne pas accumuler de retard. Finalement, n'hésitez pas à poser des questions sur la gestion du projet ou tout autre aspect technique.

Partage du travail d'équipe

Même si le projet doit être réalisé en équipe, il est essentiel de garder une forme de séparation pour certaines parties du projet. Plus spécifiquement, ce projet s'inscrit dans un contexte académique et plusieurs considérations d'évaluation individuelle doit être considérées.

Au final, il faut être en mesure d'identifier au moins une section du projet qui est prise en charge par chaque membre de l'équipe. Ça ne veut pas dire que cette partie doit être exclusivement développée par cet étudiant mais que s'est lui qui est en charge de sa réalisation et que, même s'il a de l'aide, il maîtrise à 100% ce qui s'y trouve. Ces parties doivent compter pour environ 100 % / (n_e + 3) (où ne représente le nombre d'étudiants dans l'équipe).

Il est essentiel que les éléments de définition du projet, de *brainstorming*, de conception et de planification se passent toujours en équipe. Aussi, vous devez être en mesure de réaliser le développement d'une bonne partie du noyau en équipe. Cette division du travail sera évaluée et comptera pour une partie importante de la note finale.

Contraintes de réalisation

Calendrier et évaluations

Ce calendrier présente une synthèse des évènements importants de la session :

Semaine		Description des activités	Rencontres et remises	Pondération	Date
1	5	**Sprint 0 – Mise en place** Identification équipe et sujet Validation équipe et sujet Description Conception Planification	Formation des équipes		à la 1re semaine
			Identification du projet		à la 1re semaine
2			* Validation du projet et de l'équipe		à la 2e semaine
3			Inscription du projet		à la 2e semaine
			Mise en place dépôt GIT et partage		à la 2e semaine
4			Document de définition	2.5 %	avant la 6e semaine
			Document de conception	30.0 %	avant la 6e semaine
5			Document de planification	2.5 %	avant la 6e semaine
6	3	Sprint 1 Infrastructure	Rencontre d'évaluation mi-mandat	7.5 %	de la 9e à la 11e semaine
7			Rapport d'avancement mi-mandat	2.5 %	voir l'horaire qui sera disponible à la 3e semaine
8					
9	3	**Sprint 2** Avancement			
10					
11					
12	4	**Sprint 3** Livraison	Rapport final	2.5 %	
			* Projet (code source)	35.0 %	
13			GIT	2.5 %	avant la présentation finale
			*readme	2.5 %	dates à déterminer**
14			Rencontre finale d'évaluation**	2.5 %	
			Vidéo de présentation	5.0 %	
15			Présentation finale**	5.0 %	

*** Important** : rencontres et documents obligatoires - sinon le travail n'est pas corrigé et un échec est systématique.

** **Important** : vous devez rester disponible à la fin de la session pour ces 2 évaluations – voir les périodes de reprise se trouvant à la fin du calendrier de la session.

On remarque qu'il existe plusieurs évaluations tout au long de la session et que la pondération n'est pas exclusivement liée aux activités de développement logiciel. Autrement dit, vous devez accorder de l'importance à certains éléments non techniques de votre projet. D'ailleurs, il est possible d'obtenir une mauvaise note pour ce cours même si votre projet est techniquement impressionnant et de bonne qualité.

Contraintes

Contraintes générales

Vous devez respecter ces contraintes spécifiques sur l'organisation de votre travail.

- Vous devez respecter le calendrier des activités.
- Vous devez former une équipe de deux étudiants (à l'exception d'une équipe de trois si la classe possède un nombre impair d'étudiants). Vous ne pouvez pas travailler seul.
- Vous devez valider avec l'enseignant votre équipe et votre projet au plus tard à la 2e semaine de la session.
- Le sujet de votre projet doit répondre à un niveau d'éthique élémentaire.
- Vous devez utiliser ces pratiques de travail d'équipe :
    - conception UML,
    - développement agile,
    - scrum,
    - programmation en binôme,
    - développement piloté par les tests.
- Il est attendu que vous travaillez ensemble au moins la moitié du temps. Travailler ensemble veut dire travailler sur le projet au même moment en échangeant activement. Autrement dit, il n'y a pas de contrainte sur lieu du travail et le travail à distance n'a aucune incidence sur cet aspect.
- La réalisation de ce projet repose beaucoup sur les aptitudes à résoudre les problèmes qui se présenteront. D'ailleurs, être capable d'identifier les problématiques et savoir où trouver les solutions sont des aptitudes essentielles qui doivent être développées afin d'être mieux préparé pour le marché du travail.
- Vous ne pouvez pas demander à quiconque de signer une entente de non-divulgation (« nda ») sur quel qu'aspects que ce soit. Si vous vous retrouvez dans une situation où votre projet est un secret, réalisez-le dans un autre contexte que dans celui du cours.
- Pour quelque raison que ce soit, il sera permis de changer de projet en cours de session. Néanmoins, deux aspects sont à considérer : toute la démarche d'identification, de conception et de planification demandée est à refaire. De plus, une pénalité sera appliquée pour chaque semaine suivant la 3e de la session.

  $$\text{pénalité} = 2.5\% \times (\text{semaine courante de la session} - 3)$$
- En informatique, la ligne est parfois mince entre le plagiat, l'utilisation de documents de référence, de tutoriaux, d'exemples disponibles en ligne et d'échange entre vos collèges et vos enseignants. Pour lever toutes ambiguïtés, vous devez obligatoirement mettre les références de tout ce que vous utilisez et qui n'est pas de vous :
    - peu importe l'élément : concept, design, portion de code …
    - peu importe leur provenance : web, livres, enseignants, collègues …

  Vous devez mettre ces informations à deux endroits; dans un résumé inclus à même le rapport final et là où leur usage se retrouvent :

    - dans vos documents (conception, planification ...)
    - à même votre code source sous forme de commentaires (soit dans l'en tête ou directement près des lignes de code concernées)

  Sachez que c'est la plus grande cause d'échec du cours!

Évaluations

Tout au long de la session, vous avez plusieurs documents à produire et rencontres d'évaluation avec les enseignants. Il est essentiel que ces remises et rencontres soient respectées au moment indiqué.

1. * avoir valider avec l'enseignant son projet et son équipe;
2. avoir rempli la feuille de définition de projet;
3. avoir créé un dépôt privé `GIT`, l'avoir partagé, avoir respecté la structure imposée (structure de dossiers, `.gitignore` et *`readme.md`) et avoir fait tout au long de la session un usage pertinent;
4. document de définition;
5. document de conception;
6. document de planification;
7. rencontre d'évaluation intermédiaire;
8. rapport d'avancement intermédiaire;
9. * rapport final;
10. * projet (code source);
11. rencontre d'évaluation final;
12. vidéo de présentation;
13. présentation finale.

Important : dans la liste qui précède, l'astérisque indique les documents dont la remise est obligatoire. Un échec est systématique si tous ces documents ne sont pas remis sérieusement.

Chacune de ces évaluations est décrite en détail dans les sections dédiées plus loin dans ce document.

Contraintes technologiques

Pour ce projet, vous avez peu de contraintes techniques :

- Vous devez obligatoirement utiliser `GIT` et un créer un dépôt privé mais accessible aux membres de l'équipe et des enseignants du cours. Voir la section `GIT` pour plus de détail.
- Vous devez obligatoirement utiliser une technologie fondamentale présentée dans le cadre de vos études collégiales incluant la session en cours.
- À partir de ces technologies, vous pouvez utiliser des outils qui n'ont pas été présentées en classe.

Par exemple, vous désirez faire un projet web utilisant `HTML`, `CSS` et `JavaScript` du côté client et `Python` du côté serveur et vous voulez utilisez `React`. Aucun problème, la librairie `React` est basée sur `JavaScript` et offre des outils pertinents liés.

D'un autre côté, vous considérez utiliser `ObjectiveC` pour développer une application `IOS`. Malheureusement il vous faudra considérer d'autres technologies car ces dernières ne sont pas enseignées dans le cadre de votre formation. Ceci s'applique même si vous avez des acquis personnels.

Contraintes de contenu technique

Vous avez quelques contraintes de contenu qui vise à garantir la démonstration de vos compétences techniques ainsi qu'un certain niveau de qualité de votre logiciel. Voici un résumé de ces contraintes. Elles sont décrites en détail dans les sections concernées.

- Interface utilisateur :
    - Votre projet doit offrir une interface utilisateur graphique.
    - Au minimum, cette interface doit inclure :
        - trois mécanismes de saisie d'information incluant la saisie d'un texte et d'un nombre
        - l'affichage d'au moins un item de chacun des éléments suivants :
            - texte
            - nombre formaté
            - contenu multimédia (image, musique, vidéo, ...)
- Échange de données :
    - Votre projet doit absolument échanger des informations à l'externe en entrée et en sortie.
    - Ces échanges peuvent être par :
        - des fichiers d'échange d'un format standardisé : `CSV`, `XML`, `JSON`, `INI`, ...
        - une base de données : relationnelle, non relationnelle, graphe, ...
- Qualité logicielle :
    - Vous devez utiliser au moins quatre structures de données différentes.
        - Trois de ces structures de données doivent être des outils proposés par les langages de programmation ou des librairies externes.
        - L'une de ces structures de données doit être entièrement programmée par vous.
        - Pour chacune de ces structures, vous devez faire au moins un exemple d'usage pertinent avec justification à l'appui.
    - Vous devez utiliser au moins trois patrons de conception autre que le singleton et l'itérateur.
        - Deux de ces patrons de conception peuvent être des outils proposés par les langages de programmation ou les librairies utilisées.
        - L'un de ces patrons de conception doit être entièrement programmé par vous.
        - Pour chacun de ces patrons, vous devez faire au moins un exemple d'usage pertinent avec justification à l'appui.
- Technologies spécifiques :
    - Vous devez utiliser au moins une fois les expressions régulières (`regex`) dans votre projet.
    - Vous devez produire au moins un algorithme pertinent nécessaire à votre projet.
    - Vous devez utiliser au moins une équation mathématique dans votre projet qui utilise plus que les quatre opérateurs fondamentaux (addition, soustraction, multiplication et division).

Activités préparatoires

Stratégie de sélection du sujet

Le sujet du projet

Vous pouvez choisir le sujet de projet qui vous intéresse. C'est-à-dire :

- un projet de votre cru provenant :
  - d'une idée complètement originale;
  - inspiré d'un projet existant;
- un projet provenant d'un devis externe :
  - une entreprise ou un organisme qui propose un projet;
  - un enseignant qui propose un projet.

L'important, est de valider avec l'enseignant si votre idée correspond aux standards du cours et que la somme du travail requis est raisonnable :

- un projet trop petit devra être bonifié alors qu'un projet trop long devra être réduit,
- un projet trop simpliste devra être revu de façon à démontrer un certain niveau de compétence technique alors qu'un projet trop complexe devra être simplifié.

Contrainte liée au sujet traité

Aucun projet ne peut contenir des aspects de morales ou d'éthiques inappropriés.

Conseils

Faites-vous plaisir! Choisissez un projet qui vous motive et pour lequel vous avez envie de vous investir. Tentez de mettre de l'avant au moins l'une de ces considérations :

- un sujet que vous aimé et qui touche à des intérêts personnels :
  - informatique,
  - artistique,
  - sportif,
  - à portée sociale ou environnementale,
  - documentaire,
  - éducatif,
  - technique,
  - scientifique.
- des compétences techniques que vous désirez pousser avant d'aller sur le marché du travail ou à l'université, que ce soit:
  - pour vous réapproprier des connaissances que vous avez manqué dans un cours,
  - pour explorer davantage un sujet technique qui vous intrigue,

- pour profiter des connaissances d'un enseignant avec qui vous vous entendez bien,
- pour orienter votre carrière informatique vers des compétences techniques spécifiques.

Finalement, n'oubliez pas que le cours de projet synthèse s'inscrit dans le cadre d'une session exigeante qui possède plusieurs autres cours importants qu'il ne faut pas négliger. Ainsi, il faut éviter de tomber dans le piège où le projet synthèse devient le centre de votre session. C'est faux, le projet synthèse est l'un des plus petits cours de votre session!

Formation des équipes et sélection du projet

D'abord, vous devez former vos équipes. Même si vous avez une intention particulière sur le sujet qui vous intéresse, il est important de connaître les membres de l'équipe pour s'assurer que les intérêts de chaque membre convergent dans la même direction.

Après avoir fait la formation des équipes, vous pouvez passer à l'identification du projet. Cette étape consiste à déterminer en équipe quel est le sujet de votre projet. Vous devriez être capable de répondre sommairement, mais précisément à ces questions :

- Qui sont les membres de l'équipe?
- Quel est le titre du projet? (le titre peut être temporaire)
- Quel est l'objectif principal du projet?
- Quelle forme prendra le livrable? Comment l'utilisateur pourra utiliser votre projet?
- Quels sont les principaux outils technologiques qui seront utilisés? Langages de programmation et IDE.
- Pensez-vous utiliser une base de données ou des fichiers pour échanger des données à l'externe.
- Vous avez des petites parties du projet qui doivent être prises en charge individuellement par chacun des étudiant. Quelles sont-elles? Autrement dit, quelle partie chaque étudiant devient-il responsable?

L'intention de cette étape n'est pas de connaître les détails du projet, mais d'être en mesure d'en faire une évaluation sommaire.

Validation de votre projet

Il est attendu qu'à la première semaine vous formiez votre équipe et que vous identifiez le sujet de votre projet.

N'oubliez pas que vous devez obligatoirement faire valider votre équipe et le sujet de votre projet avec l'enseignant au plus tard à la 2^{e} semaine. Cette validation passe par une rencontre informelle avec l'enseignant et tous les membres de l'équipe. Vous devez être capable de répondre aux questions présentées au point précédent.

Lors de cette rencontre, il faut être en mesure de bien cerner les objectifs que vous avez. L'enseignant se réserve le droit d'ajuster l'ampleur et la complexité de votre projet afin de le rendre réaliste et pertinent.

Si vous vous trouvez dans une impasse concernant la sélection du sujet, discutez-en le plus rapidement avec votre enseignant pour qu'il vous propose certains sujets. À la limite, l'enseignant peut vous imposer un sujet si vous êtes incapable de faire un choix.

Lorsque votre projet est validé, vous devez inscrire le projet.

Inscription du projet

Après avoir fait valider l'équipe et le projet par l'enseignant, vous devez remplir quelques informations sur un document partagé avec tous les membres de la classe. Le document est disponible ici.

Ce document présente le sommaire de tous les projets de la session :

- Titre du projet
- Membres de l'équipe
- Plateforme ciblée
- Technologies principales utilisées
- Courte description

GIT

Création du dépôt

Dans ce cours, l'utilisation de GIT n'est pas optionnelle et consiste à démontrer votre capacité à utiliser un outil de gestion de version. De plus, il serait surprenant de ne pas utiliser un tel outil en fin de formation pour un travail d'équipe.

Dès la 2e semaine, vous devez :

- créer un dépôt `GIT`
- le mettre sur un hébergeur (`GitHub` par exemple)
- il est obligatoire que votre dépôt soit privé sur l'hébergeur
- le dépôt doit être partagé avec :
    - tous les membres de l'équipe
    - les enseignants du cours

Pratique

Il est attendu que vous ayez une pratique minimalement adéquate de `GIT` :

- Concernant les `commits` :
    - `commits` fréquents appliqués sur de petits changements,
    - messages pertinent utilisant un vocabulaire standardisé.
- L'utilisation d'au moins quelques branches est attendue.
- Vous devez insérer 4 `tags` dans votre `GIT` qui représentent la fin des 4 étapes du développement. À la fin, il devra être possible de retrouver exactement ces `tags` :
    - Sprint 0
    - Sprint 1
    - Sprint 2
    - Sprint 3

Structure du dépôt

La structure du projet est imposée et doit correspondre strictement à ce qui est présenté ici :

- racine
  - .git — dossier caché
  - C61 — dossier
    - sprint0 — dossier
      - doc — dossier
      - Mandat.pdf — document à remettre
      - Conception.pdf — document à remettre
      - Planification.pdf — document à remettre
    - sprint1
      - doc — dossier
    - sprint2
      - doc — dossier
      - RapportAvancementMiMandat.pdf — document à remettre
    - sprint3
      - doc — dossier
      - RapportFinal.pdf — document à remettre
      - Présentation.mpg — document à remettre
  - dev — dossier
    - ... — tout votre développement technique
  - .gitignore — fichier
  - readme.md — fichier

Le dossier `C61` doit contenir tous les documents d'accompagnement et de documentation de votre projet. Outre le développement technique, c'est ici que doivent se retrouver tous les documents à remettre.

Les sous-dossiers `doc` sont optionnels. Leur usage est laissé à l'entière discrétion des étudiants. L'objectif est d'y retrouver tous les documents de travail qui ne sont pas à remettre spécifiquement. Par exemple, vous avez créez un document `Word` pour faire la rédaction du document `PDF` que vous remettez, c'est ici qu'il doit se retrouver. C'est la même chose pour un document de conception `UML` ou un graphique utilisé pour représenter vos maquettes. D'ailleurs, vous pouvez y créer des sous-dossiers selon la structure que vous jugez pertinente selon vos besoins.

Le sous-dossier `dev` est le coeur de votre projet. C'est ici qu'on retrouve tous les développements techniques (votre code). La structure est laissée à votre discrétion car chaque technologie utilise des pratiques différentes.

Le fichier `readme.md` est très important et obligatoire. Son contenu est décrit dans la dernière partie de ce document concernant les remises du Sprint 3.

Première partie - Mise en place

Document de définition

Contextualisation

Puisque tous les projets sont à sujet ouvert, vous devez déterminer en détail quels sont les tenants et aboutissants du vôtre.

Après avoir fait accepter votre projet par l'enseignant, vous devez présenter dans le détail quels sont les attentes du client. Ainsi, vous devez porter le chapeau du client et spécifier quels sont les objectifs spécifiques à accomplir.

Ayant le chapeau du client, vous pouvez considérer ce document comme une présentation sommaire des objectifs du projet que vous destinez à un professionnel pour qu'il en fasse le développement.

Pour ceux qui s'engageront dans la réalisation d'un projet venant de l'industrie, il est important de valider l'interprétation que vous pouvez faire d'un tel devis. Cette validation avec les personnes qui l'ont rédigée permet d'éviter, dès l'amorce du projet, la production d'une application qui diffère des intentions initiales. Dans tous les cas, vous devez reproduire ici un mandat ajusté aux détails du cours. Il peut être intéressant de mettre en annexe le mandat original s'il existe.

Document à produire

Vous devez produire un document de définition de projet. Ce document devient le mandat que vous désirez entamer pendant la session.

Le cours de projet synthèse vise à réaliser un projet dans un contexte similaire de l'industrie. Ainsi, l'objectif est de figer quels sont les éléments techniques de ce projet et quelle est votre intention au départ. C'est le mandat que vous pourriez recevoir d'un client.

Ce document doit présenter ces éléments :

- Page titre typique d'un travail académique incluant le titre officiel du projet et l'identification des membres de l'équipe
- Présentation générale :
    - Vous présentez de façon très concise quel est l'objectif principal du projet.
    - Entre 1 et 3 phrase maximum.
- Présentation détaillée :
    - Vous devez être en mesure de donner les détails du projet.
    - Le client doit identifier le contexte applicatif et quelles sont les fonctionnalités attendues.
    - C'est ici qu'on répond à toutes les questions d'orientation du projet. Par exemple, vous faites un jeu de type « *Tower Defense* » :
        - quel est le scénario (espace, far west, …),
        - combien de types de tours différentes et d'ennemis différents,
        - pour chacun d'entre eux,

                - quelles sont leurs caractéristiques détaillées,
                - quelle est la règle qui permet de les faire évoluer et quelles sont les évolutions
            - est-ce un jeu à chemin ouvert, chemin déterminé, chemin aléatoire
            - ...
        - Cette partie peut être plus ou moins longue selon la nature de votre projet. Typiquement, on peut retrouver ici un document variant de 3 à 6 pages.
- Contraintes applicatives du projet :
    - Vous devez présenter les contraintes liées à la fonctionnalité de votre projet. Par exemple : tel module est essentiel et tel autre optionnel, vous devez avoir au minimum 30 FPS, votre site requiert la notion d'usagers et la date de naissance de ce dernier ...
    - Vous devez identifier au moins deux contraintes.
- Quelle est la plateforme ciblée? Web, Windows ou Android?

Ce document ne doit pas être technique avec des détails sur la réalisation, mais plutôt une présentation pratique du projet que vous vous apprêtez à produire. Ainsi, vous ne devez pas avoir un vocabulaire technique. À la limite, considérez que vous vous adressez à un étudiant de première session en informatique. Votre lecteur possède quelques notions de base, mais sans plus.

Document de conception

Contextualisation

Avant d'amorcer la réalisation technique du projet, il est essentiel de prendre un peu de recul et de définir la démarche ainsi que les stratégiques technologiques envisagées.

Cette étape est cruciale et permet, grâce à un effort de réflexion et d'abstraction, de déterminer quelle est la vision technique du projet. Puisqu'il vous est demandé de produire un projet d'un certain niveau technique il est essentiel de s'organiser pour ne pas simplement produire pour produire. Ainsi, il faut trouver l'équilibre entre la productivité à court terme (les livrables immédiats) et la productivité à long terme (la réutilisabilité, la maintenance et l'amélioration du projet).

Comme tous les projets de développement, les objectifs sont déterminés à l'avance, mais ne sont pas définitifs et immuables. Toutefois, ils permettent de fixer une ligne directrice efficace et cohérente. Il est fort probable, voire inévitable, que plusieurs changements viendront en cours de développement. Sachez qu'il tout à fait naturel de faire évoluer le produit en fonction des changements de contraintes, du mandat de la part du client et des stratégies venant de l'équipe de développement. D'ailleurs, cette ligne de pensée est implicite avec le développement Agile.

Dans cet esprit, les premières semaines de la session doivent servir à réfléchir sur les stratégies à déployer et à préparer le développement à venir. On vous demande de produire un document qui présente une conception étayée, approfondie et détaillée du projet. Ce document doit servir de référence pendant la production.

Directement, ce document devient le devis technique du développement du projet. L'usage d'un tel document joint à la méthode de gestion de développement *Agile*, permettra de mettre toutes les chances de votre côté afin d'atteindre les objectifs que vous vous êtes fixés.

Niveau technique attendu

Ce projet est une excellente occasion de mettre de l'avant les notions de conception plus avancées et de pousser une plus grande abstraction du développement. Dans ce cours, ce n'est pas le nombre de fonctionnalités qui est évalué mais plutôt la qualité de l'infrastructure logicielle développée. L'objectif étant de donner de la valeur à votre développement et que vous soyez fier de le présenter à divers employeurs potentiels. Gardez en tête que le développement logiciel coûte cher et qu'un employeur valorise grandement ces considérations du génie logiciel : modularité, réutilisabilité, flexibilité, déploiement, maintenance, ...

Par conséquent, il est attendu que vous fassiez un effort particulier pour approfondir les aspects techniques de votre développement afin d'exploiter des notions de conception plus avancées. On parle ici des considérations sur les :

- données et services
- structures de données internes et externes
- algorithmes
- différents paradigmes de programmation dont l'orientée objets et ses constituants
- patrons de conception et architectures logicielles
- éléments de l'ergonomie logicielle
- stratégies de gestion de projet

Par exemple vous pouvez démontrer vos compétences avec les notions d'encapsulation, d'héritage et de polymorphisme. Puisque certains projets se prêtent moins bien à une structure orientée objets, il est tout de même important de pousser des concepts équivalents dans les autres paradigmes de programmation.

Gardez en tête que le niveau d'abstraction mis de l'avant dès votre phase de conception sera l'un des points techniques évalués les plus importants.

Contraintes de contenu

N'oubliez pas que vous devez démontrer votre effort de réflexion par la réalisation de plusieurs éléments techniques spécifiques. Le cours de projet synthèse sert aussi d'opportunité pour consolider toutes vos connaissances et de faire quelques explorations par vous-même.

On vous demande de faire l'analyse et l'exploitation de 7 éléments techniques de plus haut niveau dans votre projet.

Interface utilisateur

Votre projet doit offrir une interface utilisateur graphique. Au minimum, cette interface doit inclure :

- trois mécanismes de saisie d'information incluant la saisie d'un texte et d'un nombre.
- l'affichage d'au moins un item de chacun des éléments suivants :
    - texte
    - nombre format
    - contenu multimédia (image, musique, vidéo, ...)

Données externes

Votre projet doit absolument échanger des informations à l'externe en entrée et en sortie. Ces échanges peuvent être réalisés par :

- des fichiers d'échange d'un format standardisé : CSV, XML, JSON, INI, ...
- une base de données : relationnelle, non relationnelle, graphe, ...

Peu importe l'approche que vous envisagez, vous devez être précis et identifier quelles sont les données et comment vous allez les organiser.

Structures de données

Vous devez utiliser au moins quatre structures de données différentes dans le projet.

- Trois de ces structures de données doivent être des outils proposés par les langages de programmation ou des librairies externes.
- L'une de ces structures de données doit être entièrement programmée par vous. Son interface de programmation doit être orientée pour vos besoins.
- Pour chacune de ces structures, vous devez faire au moins un exemple d'usage pertinent et justifier pourquoi cette dernière est le bon choix.

Patrons de conception

Vous devez utiliser au moins quatre patrons de conception différents dans le projet.

- Trois de ces patrons de conception peuvent être des outils proposés par les langages de programmation ou des librairies externes.
- L'un de ces patrons doit être entièrement programmé par vous. Son interface de programmation doit être orientée pour vos besoins.
- Pour chacun de ces patrons, vous devez faire au moins un exemple d'usage pertinent et justifier pourquoi ils sont le bon choix dans le contexte.

Expressions régulières

Vous devez utiliser une expression régulière dans votre projet. Cette expression régulière doit être en mesure de valider une chaîne de caractères en entrée et d'extraire une sous-chaîne d'information (« capture »).

C'est à vous d'identifier à quel endroit cette information est pertinente.

Algorithme

Vous devez produire un algorithme pertinent à votre projet. On parle ici de produire une démarche procédurale qui vise à résoudre un problème spécifique. Vous pouvez produire un algorithme inédit ou implémenter une solution à un problème connu.

Toutefois, gardez en tête que vous devez faire la démonstration que vous comprenez tous les détails de ce que vous remettez.

Vous trouverez ici plusieurs exemples d'algorithmes.

Mathématique

On vous demande de démontrer un niveau de compétence minimal en mathématique. Ainsi, vous devez utiliser et implémenter une équation mathématique qui dépasse les quatre opérateurs fondamentaux (addition, soustraction, multiplication et division).

Sachez que pour la majorité des projets, vous aurez l'impression que les mathématiques sont inutiles et ne s'appliquent pas. Toutefois, gardez en tête que les mathématiques sont pertinentes partout et il vous appartient de le déterminer pour votre projet.

Vous pouvez considérer que les algorithmes possèdent souvent des considérations mathématiques et qu'il est possible de fusionner ces deux requis en un.

Document à produire

Vous devez produire un document de conception étayé qui présente la réflexion approfondie que vous avez faites du projet.

Vous devez réaliser que cette partie du projet est certainement la plus difficile.

Ce document doit présenter :

- Page titre typique d'un travail académique incluant le titre officiel du projet et l'identification des membres de l'équipe
- Toutes les maquettes de l'interface graphique – détaillées.

- Conception UML :
    - Diagramme(s) des cas d'usage – détaillé(s)
    - Diagramme(s) de classes – détaillé(s)
- Schéma(s) de la structure de données externe – détaillé(s)
- Éléments de conception :
    - Expliquer quels sont vos choix, quelles sont vos motivations pour ces choix et quels sont les éléments techniques propres à chacun :
        - structures de données
        - patrons de conception
        - expression régulière
        - algorithme
        - mathématique

Sauf pour la dernière partie, votre travail se résume à produire plusieurs diagrammes et schémas illustrant vos efforts de conception. Il est important de comprendre que vous devez produire ces schémas et diagrammes dans d'autres logiciels et assembler le tout en un seul document cohérent. De plus, n'oubliez pas que tous les documents de travail doivent se trouver dans le sous-dossier `/sprint0/doc/` de votre GIT.

Il est recommandé et encouragé d'utiliser un logiciel de conception ou de dessin vectoriel pour produire les graphiques demandés. Toutefois, vous pouvez toujours les faire à la main. Dans ce cas, vos schémas doivent être lisibles, propres, numérisés à une résolution suffisante et inclus dans votre document de conception. Les documents papiers originaux ne sont pas à remettre dans ce cas.

Document de planification

Vous devez produire un document qui présente votre planification de chacun des Sprints. Cette planification doit présenter la liste des tâches principales nécessaires à la réalisation de chacune des étapes associées. Ces tâches doivent viser un découpage modulaire du projet de façon à ce que chaque module amène une fonctionnalité particulière. On vous demande de produire un fichier Excel dans lequel se trouve ceci :

- Les noms des membres de l'équipe.
- Le titre de votre projet.
- pour chaque tâche (entrée dans la planification) :
    - un numéro de tâche, de 1 à n;
    - un titre formé d'une description technique sommaire de haut niveau, de 1 à 6 mots;
    - optionnellement, si le titre est ambigu, sous forme de commentaires dans la cellule de titre, une explication un peu plus détaillée (dans Excel, voir Review, New Comment dans le groupe Comments);
    - la liste des tâches préalables en indiquant les numéros de tâche;
    - une cote de priorité : essentiel, souhaité, optionnel;
    - une cote de difficulté : facile, moyen, difficile, incertain;
    - une estimation réaliste du temps requis pour la réalisation de la tâche, en minutes par intervalle de 30 minutes;
    - l'identifiant du Sprint où sera réalisée la tâche : Sprint 1, Sprint 2, Sprint 3;
- sur une deuxième feuille, un sommaire indiquant, pour chacun des sprints :
    - le nombre de tâches;
    - le temps total requis
    - la difficulté moyenne : vous devez faire une pondération moyenne selon cette formule :

      $\Sigma\, d \times t \,/\, \Sigma\, t$

      où : t est le temps (en minutes)

      d est la difficulté et vaut 1, 5, 10 et 7.5 pour facile, moyen, difficile et incertain respectivement

Deuxième partie : Sprints 1 et 2

Rencontre d'évaluation de mi-mandat

Contexte

Chaque équipe aura deux rencontres d'évaluation avec l'enseignant durant la session. Ces évaluations seront au 2/3 de la session (de la 9e à la 11e semaine) et à la toute fin.

Un calendrier sera généré à la 3e semaine pour établir l'ordre des rencontres. Sachez que les attentes sont ajustées en fonction de la semaine de la rencontre. Ainsi, il est attendu qu'une équipe passant à la semaine 9 soit moins avancée qu'une équipe rencontrée à la semaine 11.

La rencontre

Cette rencontre vise à évaluer l'avancement général de votre projet. Elle consiste en une brève rencontre où chaque équipe est jugée sur le progrès du travail réalisé. Cette évaluation est technique et davantage orientée sur votre compréhension du projet que sur le code lui-même ou d'autres considérations telles que les éléments d'apparence graphique. Cette évaluation est d'environ 10 minutes seulement (c'est très rapide!).

De façon plus précise, la forme de l'évaluation se fait en deux parties :

1. une brève démonstration du travail réalisé par l'équipe :
    - pour une équipe de 2 étudiants :
        i. durée : environ 4 minutes
        ii. étudiant 1 :
            1. présente le projet et les grandes lignes d'où vous êtes rendu — 30 secondes
            2. fait une petite démo d'un 1re cas d'usage fonctionnel — 30 secondes
            3. présente et explique un élément qu'il a lui-même réalisé — 1 minute
        iii. étudiant 2 :
            1. fait une petite démo d'un 2e cas d'usage fonctionnel — 30 secondes
            2. fait une présentation plus spécifique d'un élément qu'il a réalisé — 1 minute
            3. présente et explique un élément qu'il a lui-même réalisé — 30 secondes
    - pour une équipe de 3 étudiants :
        i. durée : environ 6 minutes
        ii. étudiant 1 :
            1. présente le projet et les grandes lignes d'où vous êtes rendu — 30 secondes
            2. fait une petite démo d'un 1re cas d'usage fonctionnel — 30 secondes
            3. présente et explique un élément qu'il a lui-même réalisé — 1 minute
        iii. étudiant 2 :
            1. fait une petite démo d'un 2e cas d'usage fonctionnel — 30 secondes
            2. présente et explique un élément qu'il a lui-même réalisé — 1 minute
            3. présente un élément technique qui a été un défi pour l'équipe — 30 secondes
        iv. étudiant 3 :

1. fait une petite démo d'un 3e cas d'usage fonctionnel 30 secondes
2. présente et explique un élément qu'il a lui-même réalisé 1 minute
3. présente sommairement où le projet et les prochains défis 30 secondes

- ne vous méprenez pas, il faut vous préparer pour cette rencontre, il faut vous assurer de faire une présentation claire, détaillée et précise dans très peu de temps;
- le temps donné est très court; si vous débordez, l'enseignant va interrompre votre présentation et vous perdrez des points pour les éléments non couverts;
- il est attendu que tous les membres de l'équipe soient prêt à l'heure convenu et qu'un environnement de travail soit chargé, disponible, fonctionnel et prêt pour la démonstration avant le début de la présentation;

2. période de questions par l'enseignant :
   - l'enseignant enchaînera avec quelques question sur les réalisations faites ou sur le travail à venir;
   - durée :
     i. pour une équipe de 2 étudiants, environ 4 minutes;
     ii. pour une équipe de 3 étudiants, environ 6 minutes.
   - chaque membre de l'équipe doit répondre tour à tour aux questions posées;
   - les questions ne seront pas orientées vers un étudiant en particulier mais si un étudiant a déjà répondu à une question précédente, alors c'est un autre étudiant qui doit répondre;
   - l'étudiant qui répond doit être en mesure d'expliquer ce qu'il propose, d'identifier dans le code où se trouve tel élément et de faire les liens avec ce qu'il reste à faire;
   - avec le peu de temps que vous avez, les réponses doivent être concises et précises.

Important : tous les membres de l'équipe doivent être présents pour cette rencontre.

Rapport de mi-mandat

à venir…

Dernière partie : Sprint 3 et remise

Avant de terminer le dernier sprint, il importe de finaliser plusieurs aspects du projet. Évidemment, le code source à remettre est à consolider et nettoyer mais il ne fait pas négliger la production de plusieurs documents. De plus, deux présentations sont à prévoir.

Préparation et finalisation du code source

Avant la remise finale, il est attendu que votre code source soit revu afin de le rendre plus présentable. Les éléments importants sont les suivantes en ordre d'importance.

Nettoyage

Vous devez retirer de votre code tous les artéfacts de test qu'il peut rester sous forme de code inutilisé ou de commentaires. Vous ne devriez pas laisser un code de production rempli de tests mis sous forme de commentaires. Cette pollution du code est viable pour de très court laps de temps et pour des tests précis. Ça ne peut pas être une façon de programmer.

Si vous ne voulez pas perdre des exemples pertinents de code, vous devriez faire des **commits** spécifiques dans **GIT** avant et après les avoir enlevés. Il est ainsi toujours possible de retrouver ces portions de code. Sinon, vous n'avez pas d'autres choix que de prendre des notes dans d'autres documents ailleurs. Évidemment, il y a des exceptions lorsqu'on peut avoir un code à haute valeur ajouté ou complexe dont on veut faire l'analyse ultérieurement. Mais ça doit rester une pratique marginale.

Références

Même si ce n'est pas le cas en entreprise, vous devez obligatoirement laisser sous forme de commentaires les références vers toutes les sources que vous utilisez. Assurez-vous d'avoir une nomenclature standardisée et de l'utiliser adéquatement à travers vos fichiers et le temps.

Par exemple, vous pourriez utiliser ce genre de standard :

```
ici ce trouve du code  # blabla : <a href="https://www.info.com/">info</a>
^                   ^  ^          ^
+- portion de code -+  |          +-- lien vers une référence sous forme de HTML
                      |
                      +----- début du commentaire
```

Ou celui-ci :

```
ici ce trouve du code  // ref : "https://www.info.com/"
^                   ^  ^        ^
+- portion de code -+  |        +-- lien vers une référence
                       |
                       +----- début du commentaire
```

N'oubliez pas que vous devez aussi inclure vos références dans le rapport final. Ainsi, si vous utilisez une norme documentaire, il sera facile de retrouver ces liens.

Documentation

La documentation reste souvent un problème car la pratique n'est pas nécessairement claire.

Dans le cadre de ce cours, voici comment il faut considérer cet aspect :

- Le code est généralement peu documenté, on s'appuie en grande partie sur l'auto-documentation. Ainsi, un effort particulier et maintenu est fait pour le nommage de tous les constituants de votre projet : types, fonctions, constantes et variables.
- La grande majorité des commentaires se retrouvent là où est définie l'Interface de programmation. Par conséquent, il importe de documenter les constituants dans cette zone pour les éléments où l'auto-documentation ne suffit pas.
- On retrouve peu de commentaire dans le code. En fait, on retrouve des commentaires seulement aux endroits où une explication est justifiée : compromis, choix technologique ou algorithmique, références externes, explication d'un choix arbitraire (souvent considéré comme un compromis).
- On ne retrouve **aucun** commentaire creux dans le code. Les commentaires creux sont ceux qui n'ajoutent rien au code source. Par exemple, ceux où on explique en français ou en anglais ce qui est écrit dans le langage de programmation. Vous êtes maintenant un.e professionnel.le, pas besoin de mentionner que le `for` fait une boucle ou que le `if` effectue une condition. D'ailleurs, ceci est considéré comme une très mauvaise pratique et est proscrite partout. Évidemment, le contexte académique est la seule exception à cette situation. Si votre code est illisible, réécrivez-le!

En-tête de fichier

Il est attendu que pour chaque fichier de code se trouve un en-tête qui doit contenir au minimum :

- le nom du fichier (introspection)
- le nom du projet
- le contexte de ce fichier, que retrouve-t-on ici? (une très courte phrase)
- le nom de l'auteur bien identifié et les autres étudiants au projet

Tentez de déterminer un standard élégant et de le respecter dans tous les fichiers.

readme.md

L'utilisation d'un fichier `readme.md` est devenu une pratique incontournable et essentielle. Ce fichier permet, pour celui qui le consulte, d'effectuer une prise de contact rapide et efficace avec votre projet.

Il existe un grand nombre de considérations et de pratiques pour la création d'un tel fichier. Cependant, vous devez obligatoirement créer ces sections :

- Titre
- Sommaire
  Brève présentation du projet.
- Installation
  Instructions nécessaires à la mise en place de l'infrastructure de développement (langage, librairie, IDE, …). Ces informations sont destinées à un programmeur qui désire utiliser votre projet dans un contexte de développement.
- Utilisation
  Cette section indique comment démarrer votre projet et quels sont les usages fondamentaux (ici, sans être nécessaires, des captures d'écrans sont généralement appréciées). Ce ne sont pas les instructions destinées aux programmeurs mais plutôt aux utilisateurs.
- Références
  La liste des références utilisées pour la réalisation de votre projet (avec liens hypertextes lorsqu'ils s'appliquent).
- Contact (section optionnelle)
- Remerciements (section optionnelle – par exemple, un enseignant autre que celui du cours qui vous a aidé)
- Licence (section optionnelle)

Important : La section Installation, décrite plus haut, doit être complète et détaillée pour que l'enseignant puisse monter une plateforme test rapidement et sans encombre. Cette partie de la remise n'est pas optionnelle, elle est obligatoire pour que votre projet soit évalué. Même si vous décidé de ne pas faire de `readme.md`, vous devez tout de même produire cette partie. Votre fichier `README` doit se trouver à la racine de votre `GIT`.

Voici quelques sources intéressantes sur des bonnes pratiques de création d'un `README` :

- Best-README-Template
- Make a README
- awesome-readme

Le format `MD` « `Markdown` » standardise simplement des éléments de mise en forme qui se lisent aussi bien en mode texte qu'avec un interpréteur. De plus, il permet l'ajout d'éléments non textuels, notamment : des liens hypertextes et des images. Vous trouverez ici une excellente référence sur le sujet.

L'usage du format `MD` est **obligatoire**.

tags

Dans votre `GIT`, il doit être possible de retrouver quatre `tags` qui indiquent **la fin** de ces activités :

- Sprint0
- Sprint1
- Sprint2
- Sprint3

Rapport final

à venir…

Vidéo de présentation

à venir…

Rencontre finale avec l'enseignant

À venir…

Présentation en classe

À venir…

Autres considérations

Abandon du cours

Si, pour une raison quelconque vous devez abandonner le cours, gardez en tête ces considérations :

- Lorsque vous reprendrez le cours, vous ne pourrez plus faire, refaire ou poursuivre le même projet. Vous devrez choisir un projet différent. Par conséquent, tout le travail de définition, de conception et de planification sera à refaire.
- Vos coéquipiers devront obligatoirement discuter avec l'enseignant pour définir une stratégie de poursuite du projet. Les avenues possibles sont :
    - rejoindre une autre équipe sous certaines conditions,
    - poursuivre le même projet en ajustant les livrables.

Si vous vous trouvez dans une situation personnelle qui vous force à abandonner le cours, il est important d'en discuter rapidement avec les membres de l'équipe et l'enseignant afin de prendre les modalités qui minimisent l'impact de cette situation.