

GNG2501

Mise à jour du progrès du projet de conception

GROUPE FA2-4

Soumis par:

IBRAHIM CAMARA, 300354920

RAHIMA DAHER, 300287494

HAMZA EL MENDRI, 300358491

AMINE BENHADDA, 300319706

LEVY COULIBALY, 300330412

22 octobre 2024

Sommaire

Liste des figures

Diagramme de Gant.....	20,31
Flow Chart du Code.....	21

Liste des tables

Table 1. Acronymes	v
Table 2. Glossaire	v
Table 3. Résultat triple et leurs impact positif et négatif	2
Tableau 4. Liste de besoins/problèmes du client	10
Tableau 5. D'étalonnage des produits	12
Tableau 6. Des spécifications cibles	13
Tableau 7. Des concepts	14
Tableau 8. Matrice décisionnelle	17
Tableau 9. Les spécifications détaillées des facteurs CPX	21
Tableau 10. Temps estimé pour les taches spécifiques.	25
Tableau 11. Hypothèse de produit critique.....	26
Tableau 12. Noms de matériel détaillés.....	28

Liste des acronymes et glossaire

Table 3. Acronymes

Acronyme	Définition
ACV	Analyse du Cycle de Vie
CPX	Conception pour l'Expérience
CO2	Dioxyde de Carbone
NDM	Nomenclature des matériels
GNG	Génie Général

Table 4. Glossaire

Terme	Acronyme	Définition
Analyse du Cycle de Vie	ACV	Processus d'évaluation des impacts environnementaux d'un produit tout au long de sa durée de vie.
Conception pour l'Expérience	CPX	Approche de conception visant à optimiser l'expérience utilisateur d'un produit.
Dioxyde de Carbone	CO2	Gaz à effet de serre produit par la combustion de combustibles fossiles, contribuant au réchauffement climatique.
Note de Modifications	NDM	Document détaillant les changements apportés à un projet ou produit.
Génie Général	GNG	Cours ou programme couvrant les principes fondamentaux du génie, généralement suivi dans les premières années d'études en ingénierie.

Introduction

Dans le cadre du de GNG2501, nous avons pour but d'améliorer l'accessibilité d'une plateforme de musique connue sous le nom de Musescore. Et pour se faire, nous allons travailler sur différents aspects dans ce document-ci. Premièrement, nous allons commencer par le rapport développement durable et CPX, ensuite nous allons définir le problème et développer le concept et mettre un plan de projet, finalement, nous allons travailler sur un concept détaillé et (NDM). Le but général de ce document est de connaître le progrès petit à petit fait durant ce projet, et aussi avoir une idée claire sur le produit que nous allons présenter.

2 Rapport de développement durable et CPX

2.1 Rapport de développement durable

Table 5. Résultat triple et leurs impact positif et négatif.

Résultat triple	Impact positif	Impact négatif
Social	Accessibilité accrue : Le produit permet aux personnes non-voyantes d'accéder à la musique et à l'apprentissage musical, ce qui favorise leur inclusion sociale.	Dépendance technologique : Les utilisateurs peuvent devenir dépendants de l'outil et de la technologie, limitant leur capacité à utiliser d'autres systèmes non adaptés.
	Autonomie des utilisateurs : Grâce à des fonctionnalités telles que la navigation par raccourcis clavier et la lecture en braille, les utilisateurs peuvent travailler de manière indépendante sans l'aide d'autres personnes.	Inégalité d'accès : Les personnes dans des régions avec un accès limité à des ordinateurs performants ou à Internet peuvent être exclues de l'utilisation du produit.
	Promotion de la créativité artistique : Le produit donne aux personnes non-voyantes la possibilité de composer, éditer et interpréter de la musique, leur offrant une plateforme pour exprimer leur talent.	Manque de formation spécialisée : Certains utilisateurs pourraient avoir besoin d'une formation spécifique pour maîtriser l'outil, ce qui pourrait constituer une barrière supplémentaire.
Environnemental	Dématérialisation des partitions : Le produit réduit l'utilisation de papier en permettant la lecture et l'édition numériques des partitions, contribuant ainsi à la diminution des déchets.	Consommation d'énergie : Le produit nécessite l'utilisation prolongée d'ordinateurs et d'autres appareils électroniques, ce qui augmente la consommation d'électricité et l'empreinte carbone.
	Réduction des déplacements : Le travail à distance, rendu possible par le produit, diminue les besoins de transport pour les cours ou les répétitions en personne, réduisant ainsi les émissions de CO ₂ .	Obsolescence des équipements : La nécessité d'avoir des ordinateurs performants pour faire fonctionner le produit peut inciter à la mise à jour fréquente de matériel, contribuant ainsi à l'augmentation des déchets électroniques.
	Optimisation des ressources numériques : En centralisant les outils de création et d'édition musicale, le produit réduit la nécessité d'acquérir du matériel supplémentaire, limitant ainsi la consommation de ressources matérielles.	Impact des serveurs : Si l'outil fonctionne en ligne, l'utilisation de serveurs pour héberger les données et services augmente la consommation énergétique et les émissions liées aux centres de données.

Économique	Gratuité de l'outil : L'outil est accessible gratuitement, ce qui permet à un large éventail d'utilisateurs, y compris ceux à faible revenu, d'y accéder sans coût initial.	Coûts de maintenance: Bien que le produit soit gratuit pour les utilisateurs, les développeurs doivent prendre en charge les coûts de maintenance, de mise à jour et de support technique.
	Création d'opportunités professionnelles : Le produit permet aux musiciens non-voyants de développer leurs compétences, créant ainsi des opportunités d'emploi dans le domaine de la musique.	Compétition avec des solutions payantes : En étant gratuit, le produit peut entrer en concurrence avec des logiciels commerciaux, ce qui peut affecter la viabilité des solutions payantes dans le marché.
	Soutien à l'économie locale: En formant les utilisateurs à l'édition et à la création musicale, le produit pourrait encourager l'émergence de projets artistiques locaux et de petites entreprises.	Formation coûteuse : Bien que l'outil soit gratuit, les utilisateurs ou institutions peuvent devoir investir dans des formations pour maîtriser toutes les fonctionnalités du produit

Pour définir les quatre étapes de l'analyse de cycle de vie (ACV) pour un produit en utilisant un produit similaire existant, nous pouvons structurer notre réponse comme suit :

Objectif et champ de l'étude:

L'objectif de cette étape est de définir le but de l'ACV et le cadre de l'analyse. Pour notre produit, un outil d'accessibilité musicale pour les personnes non-voyantes, l'objectif serait de mesurer son impact social, environnemental, et économique afin d'évaluer ses bénéfices et ses risques tout au long de son cycle de vie. Le champ de l'étude inclut l'analyse de l'ensemble des étapes de production, d'utilisation et de fin de vie, en prenant en compte les aspects de fabrication du matériel (ordinateurs, périphériques), utilisation du logiciel, consommation énergétique et recyclage des équipements.

Analyse de l'inventaire:

Cette étape consiste à collecter toutes les données sur les flux de matières et d'énergie associés à notre produit. Par exemple, pour l'outil musical accessible, l'analyse inclurait :

- Les ressources nécessaires pour le développement et la maintenance du logiciel.
- La fabrication des ordinateurs ou appareils requis pour utiliser l'outil (énergie, matières premières).

- Les flux énergétiques liés à l'utilisation prolongée de ces appareils.
- Les émissions de CO₂ liées au transport et à l'hébergement de données (serveurs).
- La consommation de papier évitée grâce à la dématérialisation des partitions.

Évaluation de l'impact:

L'évaluation de l'impact quantifie les effets environnementaux, sociaux et économiques identifiés lors de l'étape précédente :

Impact environnemental: réduction de l'utilisation de papier, mais augmentation de la consommation d'énergie et production potentielle de déchets électroniques (obsolescence des équipements).

Impact social: accessibilité accrue pour les non-voyants, autonomisation des utilisateurs grâce à des fonctionnalités adaptées, mais risque d'inégalités d'accès selon les ressources disponibles dans chaque région.

Impact économique: opportunités de formation et d'emploi pour les non-voyants, coût nul pour les utilisateurs, mais nécessité de couvrir les coûts de maintenance pour les développeurs.

Interprétation:

Cette dernière étape consiste à analyser les résultats obtenus, identifier les domaines critiques et proposer des améliorations. Pour ce produit :

Points positifs: favorise l'inclusion sociale, réduit l'utilisation de papier, et crée des opportunités professionnelles.

Points négatifs: dépendance technologique, obsolescence des équipements, consommation énergétique accrue.

L'interprétation permet de recommander des solutions telles que l'optimisation énergétique du produit, l'intégration de technologies plus durables, ou la formation des utilisateurs pour maximiser l'impact positif tout en réduisant les effets négatifs identifiés.

2.2 Conception pour X

1. Conception pour l'accessibilité

i. Objectifs/Besoins communs :

- Assurer que l'outil est utilisable par tous et permet une participation équitable, y compris les personnes en situation de handicap (visuel, auditif, moteur, cognitif, etc.).

ii. Exemples de métriques :

- Taux de satisfaction des utilisateurs en situation de handicap (en pourcentage).
- Temps moyen nécessaire pour qu'un utilisateur handicapé maîtrise l'outil (en heures).

iii. Exemples de contraintes :

- Compatibilité multisensorielle : L'outil doit offrir des modalités alternatives pour l'audio, le visuel et le tactile.
- Normes d'accessibilité : L'outil doit être conforme aux normes d'accessibilité canadienne.

iv. Exemples de critères de conception :

- Interface utilisateur simple et intuitive avec des options de personnalisation comme la grande taille de texte et un contraste élevé.
- Compatibilité avec les dispositifs d'assistance: lecteurs d'écran, interfaces braille, etc.

2. Conception pour la gratuité de l'outil

i. Objectifs/Besoins communs :

- Fournir un accès gratuit à l'art et aux performances, sans barrière financière pour les utilisateurs finaux.
- Faciliter la diffusion et l'utilisation de l'outil par les communautés et artistes.

ii. Exemples de métriques :

- Nombre d'utilisateurs actifs de l'outil.
- Coût de développement et de maintenance (en dollars canadien).

iii. Exemples de contraintes :

- Coût limité pour le développement et la maintenance de l'outil.
- Pas de frais d'inscription ou d'achat de licences pour les utilisateurs finaux.
- Accès à des technologies abordables : Par exemple, l'outil doit fonctionner sur des ordinateurs sans nécessiter d'équipement spécialisé.

iv. Exemples de critères de conception :

- Facilité de distribution, le fichier ou l'application doit être téléchargeable et utilisable sans restriction de géolocalisation, avec un accès minimal à Internet.

3. Concevoir pour la fiabilité

i. Objectifs/Besoins communs :

- Assurer que l'outil ou le système fonctionne de manière cohérente et prévisible dans différentes conditions d'utilisation, en minimisant les défaillances, les erreurs ou les interruptions. La fiabilité permet aux utilisateurs de faire confiance à l'outil pour répondre à leurs besoins sans défaillance.

ii. Exemples de métriques :

- **Erreurs critiques** : Nombre de fois où une erreur entraîne une perte de données ou empêche l'outil de fonctionner.
- **Disponibilité** : Pourcentage de temps pendant lequel l'outil est pleinement opérationnel

iii. Exemples de contraintes :

- **Normes de sécurité** : L'outil doit respecter certaines règles de sécurité
- **Tolérance aux pannes** : L'outil doit continuer à fonctionner, même si une partie tombe en panne.

iv. Exemples de critères de conception :

Surveillance des performances : Installer des systèmes qui vérifient en temps réel si le logiciel fonctionne correctement.

4. Concevoir pour la simplicité

i. Objectifs/Besoins communs :

S'assurer que le logiciel est simple à utiliser, surtout pour les personnes en situation de handicap particulièrement les personnes à visibilité réduite ou nul . Il doit être facile à comprendre et à naviguer, avec le moins d'efforts possibles, pour que tous les utilisateurs puissent l'utiliser sans difficulté.

ii. Exemples de métriques :

Temps de prise en main : Combien de temps, en moyenne, il à un élève aveugle pour apprendre à utiliser le logiciel.

Nombre de clics ou étapes : Combien de clics ou étapes sont nécessaires pour accomplir une tâche courante.

iii. Exemples de contraintes :

Compatibilité avec les lecteurs d'écran : Le logiciel doit être entièrement compatible avec des outils comme les lecteurs d'écran (ex. : LIME, NVDA) pour que les élèves aveugles puissent naviguer facilement.

Navigation simplifiée : Les commandes doivent être claires, avec un minimum d'étapes ou d'actions pour accomplir des tâches.

Conformité aux normes d'accessibilité : Le logiciel doit respecter les règles d'accessibilité, comme les **WCAG** (Web Content Accessibility Guidelines).

iv. Exemples de critères de conception :

- **Interface claire** : Conception d'une interface simple, avec des instructions claires et des options bien organisées pour réduire la confusion.
- **Raccourcis clavier** : Fournir des raccourcis claviers pour permettre aux utilisateurs de naviguer rapidement sans avoir besoin d'une souris.

5. Conception pour l'expérience utilisateur :

i. Objectifs/Besoins communs :

- Maximiser la satisfaction des utilisateurs : Créer une interface ou un produit qui est agréable et simple à utiliser.
- Optimiser la fluidité de l'interaction : Minimiser le nombre d'étapes ou de frictions rencontrées par l'utilisateur lors de l'utilisation de l'outil.

ii. Exemples de métriques :

- **Taux de satisfaction utilisateur** : Mesurer le degré de satisfaction des utilisateurs après l'utilisation de l'outil (sur 10).
- **Temps moyen pour accomplir une tâche** : Combien de temps un utilisateur met en moyenne pour effectuer une tâche clé (en minutes ou secondes).
- **Taux d'erreur ou de friction** : Nombre de fois où les utilisateurs rencontrent des erreurs, des blocages ou des frustrations en essayant d'accomplir une tâche.
- **Taux de rétention** : Mesurer combien d'utilisateurs reviennent régulièrement utiliser l'outil après leur première utilisation.

iii. Exemples de contraintes

- **Temps de réponse rapide** : Les actions de l'utilisateur doivent être exécutées rapidement, sans délai excessif, pour garantir une expérience fluide.
- **Compatibilité avec PC** : Le logiciel doit être entièrement compatible avec les ordinateurs, offrant une expérience fluide et cohérente sur différents systèmes d'exploitation (Windows, macOS).

iv. Exemples de critères de conception :

Tests utilisateurs : Effectuer des tests réguliers avec des utilisateurs réels pour identifier les zones de friction et optimiser l'expérience en continu.

Design simple : L'interface doit avoir un design minimaliste, avec des options clairement organisées, et éviter la surcharge d'informations.

3 Définition du problème, développement de concepts et plan de projet

1.1 Définition du problème

Tableau 4. Liste de besoins/problèmes du client

Besoins/Problèmes	Informations Connues	Informations Inconnues
Accessibilité complète pour tous les utilisateurs	Le logiciel doit être utilisable par des personnes ayant des handicaps (visuels, moteurs, etc.), compatible avec des lecteurs d'écran et des interfaces multisensorielles.	Quels sont les types d'équipements spécifiques utilisés par les personnes en situation de handicap ?
Gratuité de l'outil	L'outil doit être gratuit pour les utilisateurs finaux, avec des solutions open source privilégiées.	Existe-t-il des coûts supplémentaires ou cachés liés à l'installation ou au matériel nécessaire ?
Fiabilité et cohérence	Le logiciel doit fonctionner sans erreurs critiques, avec une surveillance des performances en temps réel pour garantir une utilisation continue.	Quels sont les facteurs environnementaux ou matériels qui pourraient affecter la fiabilité ?
Simplicité d'utilisation	L'interface doit être simple et intuitive, avec des raccourcis clavier et une navigation simplifiée pour les utilisateurs, en particulier les personnes non-voyantes.	Quels sont les besoins de formation ou de support supplémentaires pour garantir une prise en main rapide et efficace ?
Expérience utilisateur optimale	L'outil doit maximiser la satisfaction des utilisateurs avec une interface simple et une navigation fluide, compatible avec plusieurs systèmes d'exploitation (Windows, macOS, etc.).	Quelle est la diversité des systèmes d'exploitation utilisés par les utilisateurs ? Quels sont les besoins spécifiques d'optimisation pour chaque système ?
Compatibilité multisensorielle	Le logiciel doit offrir des modalités alternatives pour l'audio, le visuel, et le tactile pour accommoder différents types de handicaps.	Quels sont les capteurs ou technologies nécessaires pour une expérience tactile efficace ?

Compatibilité avec les dispositifs d'assistance	L'outil doit fonctionner avec des dispositifs d'assistance comme les lecteurs d'écran et les interfaces braille.	Existe-t-il des dispositifs spécifiques, peu courants, qui nécessitent un support supplémentaire?
Support des partitions en braille et édition musicale	Le logiciel doit permettre la lecture et l'édition de partitions en braille, ainsi que la création de partitions adaptées à un usage professionnel.	Quelle est la technologie de conversion braille la plus fiable et la plus accessible actuellement ?
Navigation simplifiée	La navigation dans l'outil doit être fluide avec un nombre minimal d'étapes pour accomplir des tâches courantes.	Quels sont les scénarios d'utilisation courants à optimiser pour réduire le nombre d'étapes nécessaires ?

Définition du problème en une phrase: Joel, ayant une situation d'handicap, trouve des difficultés à utiliser le logiciel *Muscore*, et nous, étudiant du cours GNG 2501, allons améliorer l'accessibilité de ce logiciel, permettant de créer et lire une partition, tout en pouvant la traduire en braille.

Joel musicien aveugle, souhaite que son application de création et lecture de partition soit plus accessible pour les malvoyants.

Étalonnage avec les logiciels d'éditeur de musique compétitifs :

Musecore: <https://musescore.com/>

Lime:

Sibelius: <https://www.sibelius.com/download/index.html>

Tableau 5. d'étalonnage des produits:

Légende de criticité: Les notes sont attribuées entre 3 et 9. (3 étant le moins critique et 9 le plus critique)

Metrique	Muse score	Lime	Sibelius
Intégration du lecteur d'écran	5	8	6
Support d'affichage en braille	4	7	5
Personnalisation des raccourcis	7	6	8
Navigation au clavier uniquement	6	8	7
Location Feedback for Cursor	3	9	4

Prise en charge du clavier MIDI	8	6	9
Délai de sortie de la parole (ms)	50 ms	30 ms	40 ms
Identification du nom de fichier	8	9	8



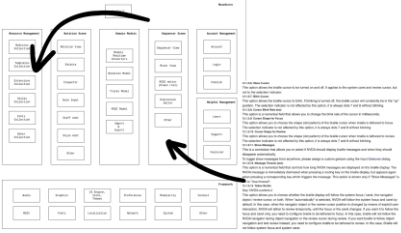
Tableau 6. des spécifications cibles:

Critères	Valeur idéale	Valeur marginalement acceptable	Raisons
Accessibilité complète	Compatibilité totale avec NVDA et interfaces braille, navigation par raccourcis clavier.	Compatibilité avec un lecteur d'écran, navigation légèrement moins intuitive.	Les utilisateurs non-voyants doivent pouvoir utiliser le logiciel facilement et efficacement.
Fiabilité et cohérence	Fonctionnement sans erreurs.	Quelques erreurs mineures, mais sans interruption du logiciel.	Les interruptions peuvent rendre le logiciel inutilisable pour les utilisateurs aveugles ou malvoyants.

Simplicité d'utilisation	Interface intuitive, raccourcis clavier, prise en main rapide.	Interface plus complexe avec raccourcis prédéfinis nécessitant une formation minimale.	Une interface trop complexe décourage l'adoption rapide par les utilisateurs.
Compatibilité multisensorielle	Retour audio, tactile (via écran braille), et visuel, avec personnalisation des options.	Compatibilité avec au moins deux modalités (audio et tactile ou audio et visuel).	Les utilisateurs avec handicaps multiples doivent pouvoir interagir via plusieurs sens.
Navigation simplifiée	Navigation optimisée avec un minimum d'étapes pour accomplir des tâches essentielles.	Navigation avec plusieurs étapes, nécessitant une courbe d'apprentissage plus longue.	Minimiser les étapes permet aux utilisateurs de rester productifs et de naviguer facilement dans l'outil.

1.2 Développement des concepts

Tableau 7. des concepts

Concepts	Explication	Sous système	Diagramme/Screenshot
Concepts 1	Plugin MuseScore rajoutant un raccourci clavier et un bouton dans la barre de navigation permettant le « playback », c'est-à-dire de faire jouer la musique à partir de la position actuelle du curseur.		
Concepts 2	API Musescore – NVDA, permettant de reconnaître l'emplacement du curseur dans une partition de musique, et de lire le tout à voix haute avec NVDA.		
Concepts 3	Modifier le programme de la partie “Resource Management - Extension collection” et ajouter une extension afin que la voix générer définie l'emplacement du curseur, la note et aussi si un dispositif braille est connectée au PC, envoyer la note sous forme braille. Et pour la partie le disposif braille, acheter un boitier existant afin de le rendre compatible avec notre programme.	Le programme original(modifiée), en C++ Boîtier braille déjà préfabriqué.	 
Concept 4	L'utilisateur utilise NVDA pour naviguer dans MuseScore. Le problème est de jouer des notes spécifiques et de connaître l'emplacement exact du curseur. Solution : Récupérer l'emplacement du curseur NVDA et comparer avec l'emplacement des notes dans MuseScore. Si l'utilisateur se trouve sur une note, un raccourci (ex. Ctrl+F) pourrait confirmer la note. Sinon, un autre raccourci pourrait indiquer la position	Intégration NVDA-MuseScore pour synchronisation de curseur et de notes	

	exacte. L'utilisateur peut aussi jouer la note via un raccourci spécifique.		
Concept 5	L'utilisateur souhaite un retour en braille des informations musicales (emplacement, son, ou note). Solution : Récupérer l'information via un raccourci et l'afficher dans un tableau braille. Ce tableau sera physiquement construit avec du carton, incluant 6 moteurs à rotation qui permettront de faire sortir ou rentrer des boutons en carton pour représenter les caractères braille en fonction de la note. Pour gérer le dispositif braille physique, une interface électrique sera mise en place à l'aide d'une carte Arduino. Les 6 moteurs à rotation seront contrôlés par l'Arduino, qui gèrera les signaux envoyés pour faire monter ou descendre les boutons en carton. Le système de translation transformera la rotation des moteurs en un mouvement linéaire vertical (montée/descente) des boutons, simulant ainsi la lecture braille.	Interface pour récupérer et traduire les informations musicales en braille + mécanisme physique de tableau braille avec moteurs et boutons en carton/Arduino pour contrôle des moteurs + système de translation de rotation en mouvement linéaire	 
Concept 6	Le concept est un MuseScore accessible pour les musiciens malvoyants, intégrant la technologie NVDA (lecteurs d'écran) et des interfaces braille. Le logiciel sera modifié pour être entièrement	Compatibilité avec NVDA : MuseScore fournit des informations vocales sur les actions en cours (lecture de notes, position du curseur). Utilisation de	

	<p>navigable au clavier, avec un retour vocal précis et optimisé.</p> <p>Le système permet également l'exportation des partitions en braille pour une lecture tactile.</p>	<p>raccourcis clavier : Facilite la navigation rapide et intuitive dans la partition.</p> <p>Sortie braille : Affiche les partitions en braille en temps réel pour une accessibilité maximale.</p>	
--	--	--	--

Tableau 8.Matrice décisionelle

	Options de concepts					
Critère de sélection	Concept 1	Concept 2	Concept 3	Concept 4	Concept 5	Concept 6
gratuité de l'outil	5	5	2	5	3	5
multisensorielle	2	2	5	2	5	4
facilité d'installation	5	5	4	5	4	4
Navigation Simplifiée	5	5	4	5	4	4
Fiabilité	3	3	3	3	3	3
Pointage Total	20	20	18	20	19	20

4. Développement de concept:

Grace aux différentes idées de concept donnée précédemment et à la matrice décisionnelle nous avons décidé d’opter pour le concept 4 comme concept de projet car c’est le concept qui répond le mieux à nos critères d’évaluation et donc aux besoins du client.

5. Représentation visuelle du concept:

(Voir tableau des concepts)

6. Rapport entre le concept et les spécifications cibles, avantages et désavantages :

Le concept repose sur l'intégration de MuseScore avec des technologies d'assistance comme NVDA et des dispositifs braille, en mettant l'accent sur l'accessibilité. Cela répond à la spécification cible d'**accessibilité complète**, car les utilisateurs aveugles ou malvoyants peuvent naviguer et utiliser le logiciel de manière intuitive via des raccourcis clavier et un retour vocal/braille. Le concept garantit également la **fiabilité et la compatibilité multisensorielle**, en offrant un retour auditif et tactile. Toutefois, la **simplicité d’utilisation** peut être compromise si la courbe d'apprentissage pour la maîtrise des raccourcis et l'interaction avec le braille est trop élevée. Un autre désavantage potentiel réside dans la complexité matérielle de l’ajout de dispositifs braille externes.

Avantages :

- Accès pour les utilisateurs malvoyants grâce à la compatibilité avec NVDA et le braille.
- Navigation optimisée par raccourcis clavier et retour vocal/braille en temps réel.
- Possibilité d'exportation des partitions en braille.

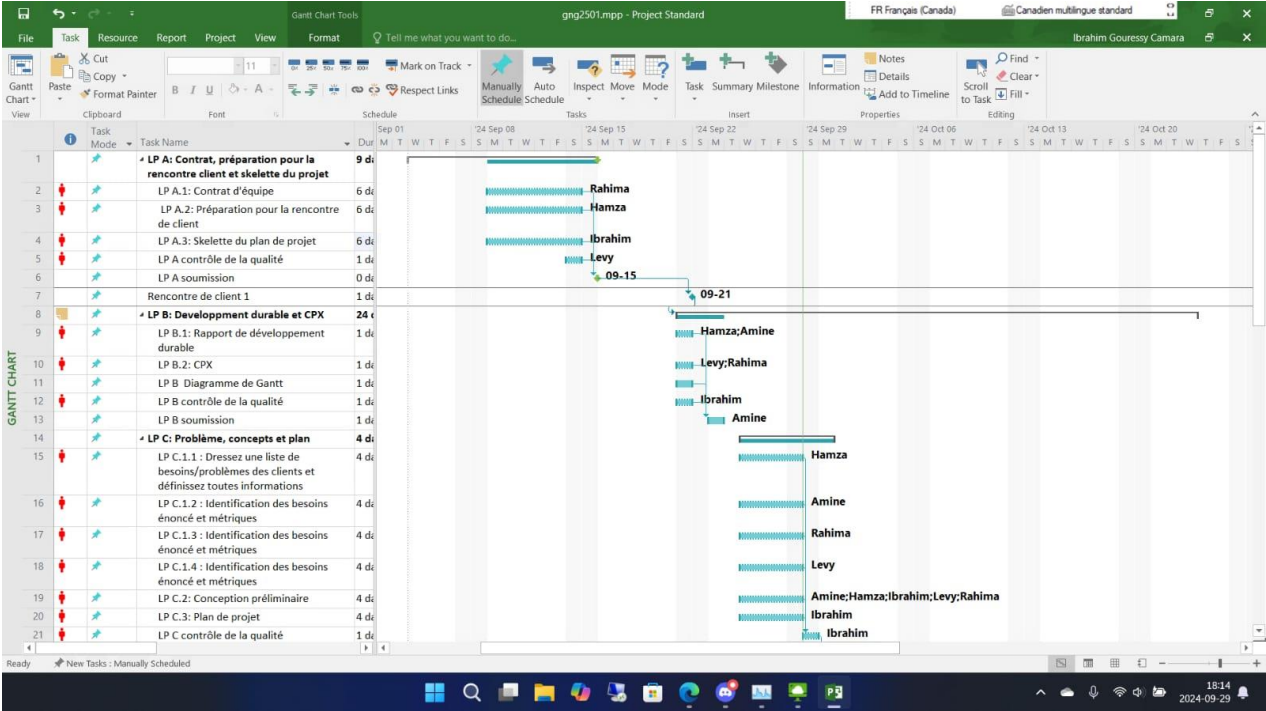
Désavantages :

1. Complexité d'utilisation pour les nouveaux utilisateurs.
2. Erreurs mineures possibles dans la synchronisation du curseur avec la lecture musicale.

7. Rapport entre le concept et les facteurs CPX identifiés dans le Livrable de projet B :

Le concept se conforme en grande partie aux facteurs CPX. Par exemple, il prend en charge **l'intégration du lecteur d'écran** (CPX:8) et la **navigation au clavier uniquement** (CPX:8), assurant une expérience fluide pour les utilisateurs non-voyants. Le **support d'affichage en braille** (CPX:7) est également bien couvert, bien que des améliorations puissent être nécessaires pour un support complet. En revanche, des limitations peuvent survenir avec le **location feedback for cursor** (CPX:3), car fournir un retour précis de la position du curseur pourrait nécessiter des ajustements techniques complexes. Le concept excelle en matière de **personnalisation des raccourcis** (CPX:6), offrant une flexibilité accrue aux utilisateurs.

3.2 Plan de projet



4 Conception détaillé et NDM

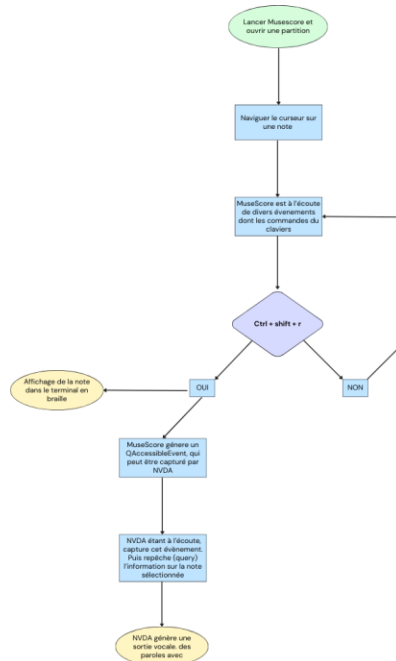
4.1 Conception détaillé

4.1.1. Résumez la rétroaction des clients reçue lors de votre deuxième rencontre client et énoncez clairement ce qui doit être changé ou amélioré par rapport à votre concept:

Lors de notre rencontre 2 avec le client, nous avons présenté deux parties pour notre concept, partie logicielle, qui répondait aux attentes du client, et aussi voulait qu'on fasse un programme short cut qui lui permettra de lui donner la position de la note. Et la partie physique, boîtier pour le retour braille, et qui elle, ne répondait aux attentes du client car le client ne voulait pas un retour braille physique mais sur le logiciel Musescore.

4.1.2 Conception détaille

Flow chart de Notre code



4.1.3 Élément à prendre en compte et CPX

Voici un tableau qui résume les éléments à prendre en compte pour concevoir le concept tout en respectant les facteur CPX.

Tableau 9. Les spécifications détaillées des facteurs CPX

Facteur CPX	Élément à prendre en compte	Importance
Conception pour l'expérience utilisateur	<ul style="list-style-type: none"> - Intégration de meilleur paramètre d'accessibilité (raccourcis clavier) - Navigation simple dans la partition 	Très élevées

Conception pour la simplicité	<ul style="list-style-type: none"> - Interface simplifiée - Raccourcis utile et facile à utiliser - Code clair et maintenable pour faciliter les modifications 	Importante
Conception pour la fiabilité	<ul style="list-style-type: none"> - Code stable (pas de bug et crash) - Compatibilité avec nvda et option braille de musecore - Performance stable pour les partitions complexes 	Moyenne
Conception pour la gratuité de l'outil	<ul style="list-style-type: none"> - Maintien du modèle open source - Optimisation des ressources pour éviter les coûts - Documentation claire pour permettre 	Importante
Conception pour l'accessibilité	<ul style="list-style-type: none"> - Adaptation complète aux personnes aveugles 	Priorité maximale

	- Bien intègre les modifications du code	
--	--	--

L'accessibilité devient la priorité principale, car c'est l'élément central pour garantir que musecore puisse être utilisée efficacement par des personnes aveugles. L'expérience utilisateur et la fiabilité sont directement liées à ce facteur. La simplicité facilite l'utilisation de musecore, alors que la gratuité assure que l'outil reste accessible à tous sans coûts.

4.1.4 Ressource et compétence

Étant donné que notre projet est entièrement logiciel, et que Musecore est un logiciel open source, nous n'avons pas vraiment besoin de ressources matérielles, et si nous rencontrons des problèmes avec le code, nous pouvons facilement utiliser YouTube et d'autres ressources en ligne pour les résoudre.

Liste de compétence et des ressources

Compétences actuelles:

Connaissance de Musecore et de son code

Maîtrise des principes de conception CPX

Travail avec des lecteurs d'écran comme NVDA

Compréhension des besoins des utilisateurs non-voyants

Compétence à développer:

Programmation en C++

Gestion de projet

Développement d'accessibilité pour MuseCore.

Ressources nécessaires:

Cours et tutoriel en C++ sur YouTube

Travailler sur gérer notre temps

Ressource sur l'accessibilité numérique

Outils de tests d'accessibilité comme NVDA, simulateurs d'accessibilité

4.1.5. Evaluation réaliste du temps requis pour mettre en œuvre le concept

Recherche et planification initiale (analyse du code, prise en main de MuseScore et NVDA) :

- Estimation pessimiste : 5 heures
- Estimation optimiste : 1.5 heures
- Estimation = $(75\% * 5) + (25\% * 1.5) = 4.25$ heures

Développement des raccourcis pour NVDA (lecture des notes et positions) :

- Estimation pessimiste : 25 heures
- Estimation optimiste : 12 heures

- Estimation = $(75\% * 25) + (25\% * 12) = 21.75$ heures

Développement des raccourcis pour le support braille :

- Estimation pessimiste : 20 heures
- Estimation optimiste : 10 heures
- Estimation = $(75\% * 20) + (25\% * 10) = 17.5$ heures

Tests de compatibilité (validation avec NVDA et dispositifs braille) :

- Estimation pessimiste : 5 heures
- Estimation optimiste : 1.5 heures
- Estimation = $(75\% * 5) + (25\% * 1.5) = 4.25$ heures

Documentation et formation des utilisateurs (rédaction de manuel utilisateur) :

- Estimation pessimiste : 8 heures
- Estimation optimiste : 3 heures
- Estimation = $(75\% * 8) + (25\% * 3) = 6.75$ heures

Estimation totale du temps nécessaire :

Étape	Temps estimé
Recherche et planification	4.25 heures
Développement des raccourcis pour NVDA	21.75 heures
Développement des raccourcis pour braille	17.5 heures
Tests de compatibilité	4.25 heures
Documentation et formation	6.75 heures

Total	54.5 heures
--------------	--------------------

Tableau 10. Temps estimé pour les tâches spécifiques.

Conclusion :

L'estimation de temps pour la mise en œuvre du projet est d'environ **54.5 heures** au total, réparti sur environ 4 semaines de travail à rythme régulier. Si des imprévus surviennent (p. ex., des bogues inattendus ou des problèmes de compatibilité), le temps pourrait être légèrement plus long.

4.1.6. Hypothese de produit critique :

Tableau 11. Hypothese de produit critique.

Hypothèse Critique	Description	Impact Potentiel	Valeurs Acceptables
Compatibilité avec NVDA	Les raccourcis doivent être compatibles avec le lecteur d'écran NVDA.	Si MuseScore ne supporte pas certaines fonctionnalités des lecteurs d'écran, l'ajout des raccourcis pourrait être limité ou peu fonctionnel.	Les raccourcis doivent être reconnus et interprétés correctement à 100 % par NVDA. La lecture vocale des notes et des positions doit être fonctionnelle et sans erreur.

Temps de réponse rapide des raccourcis	Les raccourcis doivent fonctionner sans latence perceptible pour la lecture des notes et la position dans la partition.	Des temps de réponse lents ou des retards dans la lecture pourraient nuire à l'expérience utilisateur.	Latence maximale de 0,5 seconde pour la lecture d'une note ou la mise à jour de la position du curseur par NVDA
Accessibilité du développement dans MuseScore	L'architecture du code de MuseScore doit permettre d'ajouter les raccourcis facilement sans limitations majeures dans le code.	Si l'architecture du logiciel impose des contraintes, cela pourrait rendre difficile l'implémentation des nouveaux raccourcis	Le code source doit permettre l'intégration des nouvelles commandes sans modifier les fonctionnalités principales.
Acceptabilité des raccourcis par les utilisateurs	Les raccourcis doivent être intuitifs et ne pas entrer en conflit avec d'autres commandes	Si les raccourcis sont contre-intuitifs ou entrent en conflit avec des commandes existantes, cela pourrait limiter leur	Les raccourcis ne doivent pas être en conflit avec les raccourcis existants. L'utilisateur doit

	existantes dans l'application.	adoption par les utilisateurs.	être capable de naviguer intuitivement après une courte période de formation (15 minutes maximum).
Stabilité de MuseScore avec de multiples fonctionnalités d'accessibilité	MuseScore doit rester stable et performant même avec l'ajout des nouvelles fonctionnalités d'accessibilité (raccourcis pour NVDA).	Si l'ajout de nouvelles fonctionnalités entraîne des bogues ou des dysfonctionnements, cela pourrait affecter l'expérience utilisateur globale et compromettre la stabilité de MuseScore.	Le logiciel doit fonctionner sans crashes ou erreurs après l'ajout des nouvelles fonctionnalités pour une utilisation continue.

4.2 NDM

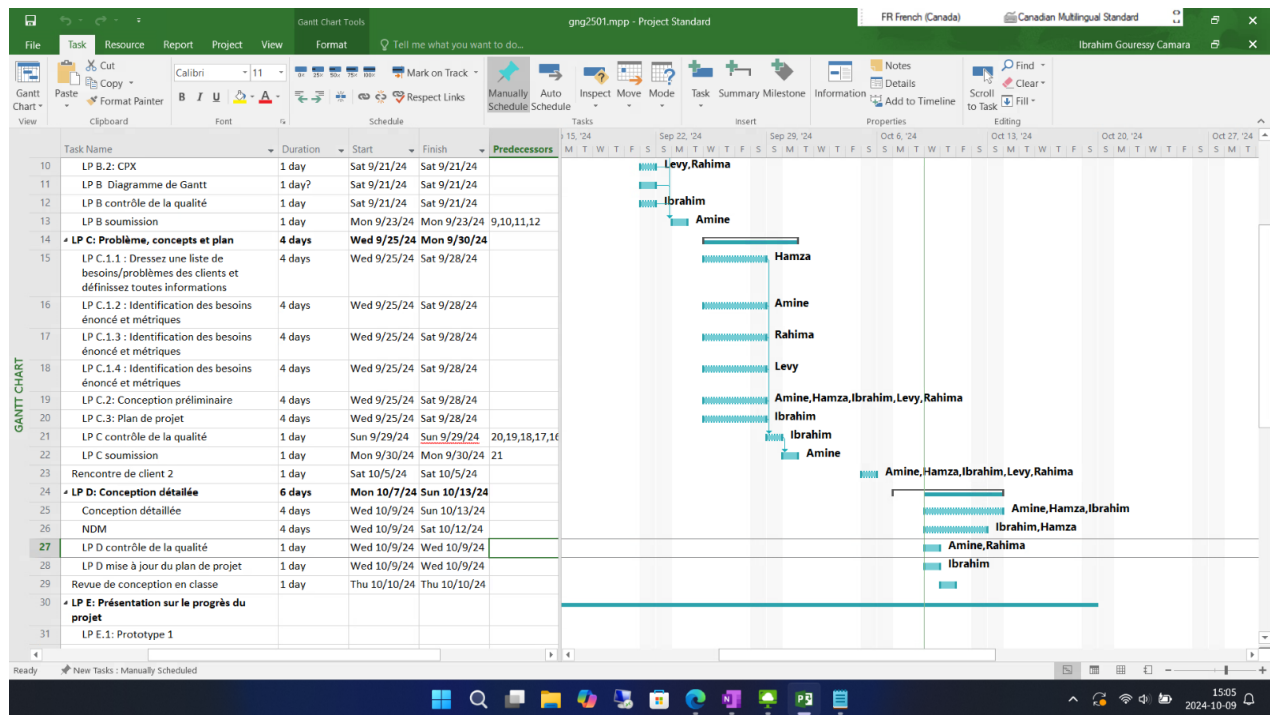
Tableau 12. Nomenclature des materiels

#	Noms de matériel	Description	Liens	Quantité	Prix/unité	Prix total
1	Github	Aplication site web de collaboration, versionnement, code, open-source, projets.	https://git hub.com/	1	0\$	0\$
2	musecore	Aplication notation musicale gratuite, accessible, collaborative.	ht tps://mus escore.co m/	1	0\$	0
3	Vs code	Site web éditeur, code, extensible, multiplateforme, productivité.	https://co de.visuals tudio.co m/	1	0\$	0\$

4	Copilot	Site web assistant, code, IA, autocomplétion, productivité.	Microsoft Copilot	1	0\$	0\$
5	ordinateur	Un ordinateur pour programmer et gérer notes projets.		1	0\$	0\$
6	Tablettes numériques	Une tablettes numérique pour concevoir et nous assister		1	0\$	0\$
7	Nvda	lecteur d'écran, gratuit, accessible, voix,	https://www.nvaccess.org/download/	1	0\$	0\$

4.3 Plan de projet

Ajouter une capture d'écran de votre diagramme de gantt.



Bibliographie

- MuseScore. (n.d.). *MuseScore GitHub repository*. GitHub. <https://github.com/musescore/MuseScore>
- MuseScore. (n.d.). *MuseScore official website*. <https://musescore.com/>
- MuseScore. (n.d.). *Developer's Handbook: Finding your way around - Git workflow*. MuseScore. <https://musescore.org/en/handbook/developers-handbook/finding-your-way-around/git-workflow>
- MuseScore. (n.d.). *Set-up developer environment*. GitHub. <https://github.com/musescore/MuseScore/wiki/Set-up-developer-environment>
- MuseScore. (n.d.). *Code structure*. GitHub. <https://github.com/musescore/MuseScore/wiki/CodeStructure>
- NV Access. (n.d.). *NVDA GitHub repository*. GitHub. <https://github.com/nvaccess/nvda>
- MuseScore. (n.d.). *MuseScore YouTube channel*. YouTube. <https://www.youtube.com/c/musescore>