Spécification des Exigences Logicielles (SEL) pour l'Application Navigateur Web (ICTU SEARCH)

1. Introduction

1.1 But du Document

Ce document de spécification des exigences logicielles (SEL) décrit les fonctionnalités et les contraintes de l'application "ICTU SEARCH", un navigateur web simple. L'objectif est de définir clairement les exigences du système pour guider le développement, les tests et la maintenance.

1.2 Portée du Produit

L'application ICTU SEARCH est un navigateur web de bureau développé avec Qt, permettant aux utilisateurs de naviguer sur internet, de gérer un historique de navigation simple (retour/avance) et de sauvegarder les URL visitées dans un fichier local. Il vise à fournir une interface utilisateur basique pour l'accès web.

1.3 Glossaire

- **URL** : Uniform Resource Locator, adresse d'une ressource sur Internet.
- **QWebEngineView** : Widget Qt pour afficher du contenu web basé sur Chromium.
- **QStack** : Conteneur Qt implémentant une structure de données de pile, utilisée ici pour l'historique de navigation.
- **QUrl** : Classe Qt pour la manipulation des URL.
- **QApplication** : Classe fondamentale pour les applications Qt avec une interface utilisateur graphique.
- **QMainWindow** : Fenêtre principale de l'application.
- **QFile, QTextStream** : Classes Qt pour la lecture et l'écriture de fichiers texte.
- **QCoreApplication::applicationDirPath()** : Fonction Qt pour obtenir le chemin du répertoire de l'application en cours d'exécution.
- **ICTU SEARCH** : Nom de l'application tel qu'affiché dans le titre de la fenêtre.

2. Description Générale

2.1 Perspective du Produit

L'application ICTU SEARCH est un système autonome, un navigateur web client. Elle utilise le moteur de rendu Chromium (via QWebEngineView) pour afficher les pages web. Elle interagit avec le système de fichiers local pour la persistance de l'historique des URL.

2.2 Fonctions du Produit

Les principales fonctions de l'application ICTU SEARCH incluent :

- **Navigation Web** : Affichage de pages web basées sur une URL saisie par l'utilisateur.
- **Historique de Navigation** : Fonctionnalités de retour et d'avance dans l'historique des URL visitées.
- **Persistance des URL** : Sauvegarde de chaque URL chargée dans un fichier texte local.
- **Interface Utilisateur Basique** : Fournit des éléments GUI pour la saisie d'URL et la navigation.

2.3 Caractéristiques des Utilisateurs

- **Utilisateur Final** : Toute personne souhaitant naviguer sur internet via une interface simple. Aucune exigence spécifique en termes de compétences techniques.

2.4 Contraintes

- **Moteur de Rendu** : Dépend de QWebEngineView, donc basé sur Chromium.
- **Persistance des Données** : L'historique complet n'est pas persisté ; seules les URL *chargées* sont enregistrées dans un fichier texte (entered_strings.txt) dans le répertoire de l'application. Les piles backStack et forwardStack sont en mémoire et sont réinitialisées à chaque lancement de l'application.
- **Stockage des URL** : Les URL sont stockées en texte brut, une par ligne.
- **Interface Maximisée** : L'application démarre en mode maximisé.
- **Chemins d'Icônes/Images** : Les chemins des icônes et des images de fond sont codés en dur dans le code (C:/Users/Lenovo/...).
- **Absence de Favoris/Onglets** : Le navigateur est un navigateur à fenêtre unique sans gestion des favoris ou des onglets.
- **Gestion des Erreurs de Réseau** : La gestion des erreurs de chargement ou de réseau n'est pas explicitement détaillée mais est implicite via le comportement de QWebEngineView.

3. Exigences Fonctionnelles

3.1 Navigation Web

- **FR1 : Saisie et Chargement d'URL**
    - Le système doit permettre à l'utilisateur de saisir une URL dans un champ de texte.

- Le système doit charger et afficher le contenu de l'URL saisie dans la vue web (QWebEngineView) lorsque l'utilisateur clique sur un bouton "Aller" (ou équivalent).
- Le système doit s'assurer que l'URL saisie est au format valide (https:// ou http://). Si ce n'est pas le cas, le système doit ajouter https:// par défaut.
- Le système doit gérer le chargement d'URL, y compris les redirections et les liens internes.

- **FR2 : Historique de Navigation**
  - Le système doit maintenir un historique des URL visitées sous forme de piles pour la navigation arrière (backStack) et avant (forwardStack).
  - **FR2.1 : Bouton "Retour"** : Le système doit permettre à l'utilisateur de revenir à la page précédemment visitée en cliquant sur un bouton "Retour". Ce bouton doit être activé uniquement si la pile "arrière" n'est pas vide.
  - **FR2.2 : Bouton "Avance"** : Le système doit permettre à l'utilisateur d'avancer vers une page précédemment visitée (après être revenu en arrière) en cliquant sur un bouton "Avance". Ce bouton doit être activé uniquement si la pile "avant" n'est pas vide.
  - **FR2.3 : Mise à jour de l'historique** : Lorsqu'une nouvelle URL est chargée (saisie directe ou navigation à partir des boutons), l'historique doit être mis à jour correctement (URL actuelle poussée sur la pile "arrière", pile "avant" effacée).

3.2 Persistance des URL

- **FR3 : Sauvegarde des URL**
  - Le système doit sauvegarder chaque URL **chargée avec succès** dans un fichier texte nommé entered_strings.txt.
  - Les URL doivent être ajoutées à la fin du fichier, chacune sur une nouvelle ligne.
  - Le fichier entered_strings.txt doit être créé dans le répertoire d'exécution de l'application.

3.3 Interface Utilisateur

- **FR4 : Affichage de la Vue Web**
  - Le système doit intégrer un widget QWebEngineView qui occupe la zone de contenu principale de la fenêtre.

- **FR5 : Affichage du Titre de la Fenêtre**
    - Le titre de la fenêtre doit être défini sur "ICTU SEARCH".
- **FR6 : Apparence Visuelle**
    - Le système doit appliquer un style visuel défini via une feuille de style (CSS) pour l'arrière-plan de la fenêtre.
    - Le système doit afficher une icône personnalisée pour la fenêtre.
- **FR7 : État des Boutons de Navigation**
    - Les boutons "Retour" et "Avance" doivent être activés ou désactivés dynamiquement en fonction de l'état des piles d'historique.

4. Exigences Non Fonctionnelles

4.1 Performances

- **NFR1.1 : Temps de Chargement de Page** : Le temps de chargement des pages web dépendra principalement du moteur Chromium sous-jacent et de la connexion réseau de l'utilisateur. Le navigateur lui-même ne devrait pas introduire de délais significatifs au-delà de ceux intrinsèques au rendu web.
- **NFR1.2 : Réactivité de l'UI** : L'interface utilisateur doit rester réactive pendant le chargement des pages web et les opérations de navigation.

4.2 Sécurité

- **NFR2.1 : Sécurité du Contenu Web** : La sécurité du contenu web est héritée de la sécurité intégrée au moteur Chromium. Le navigateur lui-même ne fournit pas de fonctionnalités de sécurité supplémentaires (ex: filtres anti-phishing, bloqueurs de pop-up).
- **NFR2.2 : Sécurité des Fichiers** : Les URL sont stockées en texte brut dans un fichier accessible localement, sans chiffrement. La sécurité de ce fichier dépend de la sécurité du système d'exploitation.

4.3 Utilisabilité

- **NFR3.1 : Interface Intuitive** : L'interface utilisateur doit être simple et directe, avec des éléments de navigation clairs (champs URL, boutons Retour/Avance/Aller).
- **NFR3.2 : Messages d'Erreur** : En cas d'échec d'ouverture du fichier de sauvegarde, un message d'erreur doit être affiché à l'utilisateur.

4.4 Fiabilité

- **NFR4.1 : Persistance des URL** : Les URL chargées doivent être systématiquement sauvegardées dans le fichier désigné.
- **NFR4.2 : Gestion des Fichiers** : Le système doit pouvoir créer le fichier de sauvegarde s'il n'existe pas et gérer les erreurs d'écriture de manière gracieuse.

4.5 Maintenabilité

- **NFR5.1 : Code Modulaire** : Le code est organisé en modules (main.cpp, mainwindow.h, mainwindow.cpp) avec une séparation des préoccupations (initialisation de l'application, logique de l'UI et fonctions utilitaires).
- **NFR5.2 : Documentation** : Les fichiers de code source contiennent des commentaires expliquant le but des classes, des méthodes et des blocs importants.

5. Exigences Techniques

5.1 Environnement de Développement

- **NFR6.1 : Langage de Programmation** : C++
- **NFR6.2 : Framework GUI** : Qt 5.x (avec modules Widgets, Core, Gui, WebEngineWidgets)
- **NFR6.3 : Outil de Construction** : QMake (implicite par la structure du projet Qt)

5.2 Stockage des Données

- **NFR7.1 : Format de Fichier** : Fichier texte (.txt) pour la sauvegarde des URL, avec une URL par ligne.
- **NFR7.2 : Emplacement du Fichier** : Le fichier entered_strings.txt sera situé dans le répertoire d'exécution de l'application.