CÉGEP SHAWINIGAN
Du savoir et des gens

Version H24

420-W1R-SW - Projet 2 - Dashboard

Type de travail : Équipe
Pondération : 25%

Délai accordé et remise

Délai accordé

Les étudiants auront jusqu'au 10 avril pour compléter ce travail. Tout retard entraînera une pénalité de 10% par jour, conformément à la politique du département.

Remise

Le projet doit être remis via Git. Vous devez faire des commits régulièrement afin de montrer l'évolution du projet. L'enseignant doit avoir accès au dépôt Git au moment de la correction.

Conseils avant de débuter

Prenez le temps de **bien lire et comprendre le(s) énoncé(s)**.

Le projet étant de plus grand envergure, vous devriez :

- **Faire fonctionner un dashboard de base** avant d'ajouter des personnalisations et de la gamification.
- Commencer avec **1-2 modules essentiels** avant d'ajouter les autres.
- Fixer des **jalons et échéances clairs** pour éviter de vous perdre dans l'optimisation.

Il est beaucoup plus important de livrer des éléments de qualités, mais un projet incomplet que de livrer un projet complet, mais bourré d'erreurs.

CÉGEP SHAWINIGAN
Du savoir et des gens

Règles et directives

- C'est un travail à effectuer en équipe de 2, mais il est également possible de l'accomplir seul.
- Le code en français ou en anglais est accepté, mais vous devez être CONSTANTS dans votre choix. Si vous codez en français, faites le partout. Si vous codez en anglais, faites le partout.
- Indiquez toujours vos références lorsque c'est possible. Par exemple, si vous trouvez un snippet de code sur internet, vous devez en indiquer la provenance en commentaires et expliquer son fonctionnement détaillé.
- Vous devez **respecter les standards et conventions**! Exemples :
  - Code bien indenté
  - Code uniforme (Ex: Code appliqué de la même façon dans la solution)
    - Sauts de lignes
    - Tabulations vs espaces
  - Nomenclature du bon format (PascalCase, camelCase, ...)
  - Nomenclature adéquate (Noms appropriés et pertinents)
    Ex. : "Toto" pour un compteur n'est pas très pertinent.

BON SUCCÈS

Objectif

L'objectif est de développer une **application web de type "Dashboard personnel"** qui servira de **page de démarrage** pour l'utilisateur, remplaçant Google ou toute autre page d'accueil.

Le projet doit permettre d'afficher une page imitant Google et permettant de lancer des recherches sur celui-ci, tout en ajoutant des modules **dynamiques** (widgets) indépendants et personnalisables.

Ce Dashboard devra être **hautement configurable**, sans backend, en sauvegardant uniquement les données localement via **LocalStorage, SessionStorage ou IndexedDB** selon ce qui est le plus pertinent selon les cas.

L'accent sera mis sur une interface moderne, fluide et interactive, intégrant un **système de gamification des objectifs et tâches** avec des niveaux, gains d'XP et pénalités.

NB. Ce projet vise également à pouvoir vous servir pour vos portefolio que vous pourrez présenter aux entreprises. Donc vous devriez porter une attention sérieuse à l'expérience utilisateur, à la fluidité et l'ergonomie.

Exigences techniques

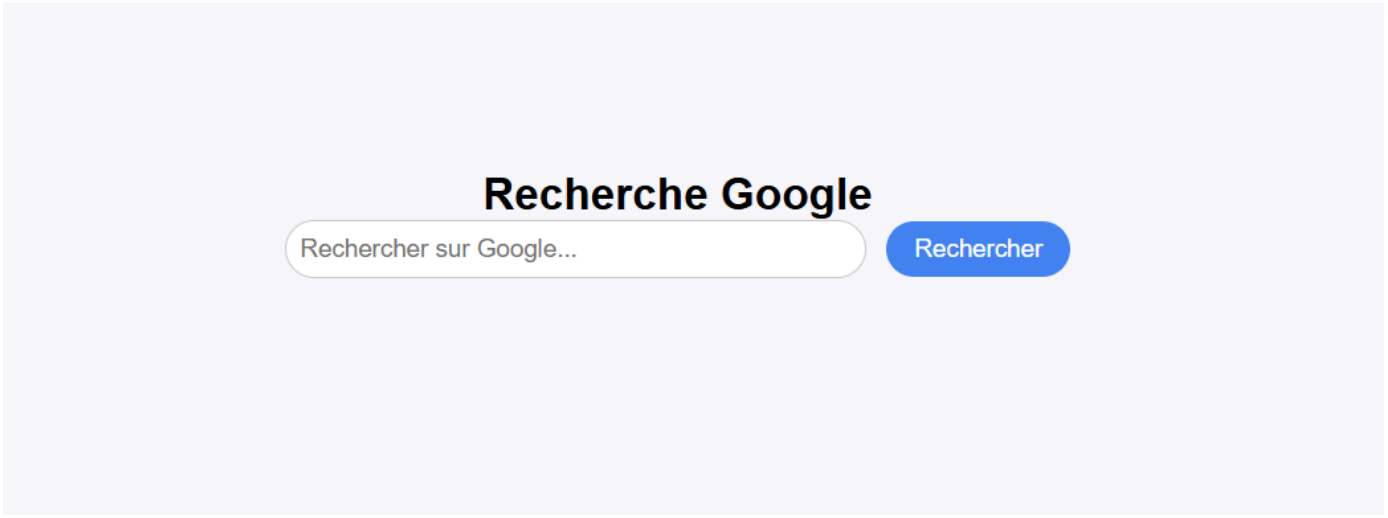
- **Technologies autorisées** : HTML, CSS, JavaScript pur (sans framework comme React, Vue, etc.).
- **Librairies suggérées** : TailwindCSS (pour le design), Chart.js (pour les statistiques, facultatif), Dragula.js (pour le drag & drop).
- **Expérience utilisateur** : Animations, transitions et interactions fluides.
- **Personnalisation avancée** : L'utilisateur doit pouvoir modifier l'apparence visuelle du Dashboard et de chacun des modules (thème, couleurs, disposition, dimensions, etc.) L'apparence doit également conservée.
- **Gestion de l'overlay** : Une combinaison de touches doit permettre d'afficher/masquer l'overlay contenant les modules.

Structure du Dashboard

Le Dashboard fonctionnera comme une page d'acceuil à votre navigateur comme Google (ou une autre page de démarrage).

L’utilisateur pourra entre autre :

- **Déplacer, redimensionner et masquer les modules indépendamment**.
- **Personnaliser les modules** (contenu, couleurs, langue, icônes, etc.).
- **Sauvegarder automatiquement ses préférences** (position, taille, affichage des modules).


Recherche Google
Rechercher sur Google...
Rechercher

Modules

Vous devez créer plusieurs modules qui seront indépendants les uns des autres.

- Individuel : Minimum 7 modules.
- Équipe de 2 : Minimum 10 modules.

Voici une liste de suggestions :

1. **Sticky Notes** (notes rapides épinglées sur le Dashboard).
2. **Liste de tâches** (To-do list quotidienne, hebdomadaire, mensuelle, etc.).
3. **Objectifs et deadlines** (suivi des objectifs à court et long terme).
4. **Suivi des habitudes** (ex : consommation d'eau, suivi de calories, etc.).
5. **Suivi du sommeil** (historique et tendances de sommeil).
6. **Suivi de l'apprentissage** (ex : suivi du temps d'étude).
7. **Calendrier avec rappels** (gestion d'événements et notifications locales).
8. **Pseudo-JIRA** (gestion simplifiée de projets et tâches en kanban).
9. **Statistiques visuelles** (piecharts, graphiques de progression).
10. **Tableau de motivation** (citations inspirantes, suivi des raisons des objectifs).
11. **Météo** (Requiert une API)
12. **Cours en bourse** (Requiert une API)

Chaque module doit pouvoir utiliser des données que ce soit sauvegardé ou en provenance d'une API.

Vous pouvez vraiment faire ce que vous voulez comme modules en autant que ce soit pertinent avec le but de l'application : un Dashboard qui puisse représenter d'un coup d'oeil la situation ou la vie d'une personne.

Personnalisation et fonctionnalités

- **Gestion des modules**: Redimensionnement, déplacement, suppression, ajout dynamique.
- **Options de thème** : Choix des couleurs, typographie, mise en page générale.
- **Mode sombre / clair** : Activation et désactivation selon la préférence utilisateur.
- **Affichage/Masquage de l'overlay** : Activation/désactivation via une combinaison de touches.

Système de Gamification

- L'utilisateur gagne de l'**XP** lorsqu'il accomplit des tâches, atteint des objectifs ou remplit des habitudes. Doit être configurable, mais rester SIMPLE.
- Des **niveaux** sont définis en fonction de l'XP accumulée.
- Des **pénalités** sont appliquées si trop d'objectifs sont manqués (perte d'XP ou affichage d'un message d'alerte). Doit être configurable, mais rester SIMPLE.
- Un tableau de bord doit afficher clairement l'évolution de l'utilisateur.

Rappels : Remise

La livraison doit se faire par un gestionnaire de source (Ex : Github (PRIVÉ)) et la structure des fichiers doit suivre les bonnes pratiques. (Html / CSS / Javascript). Vous devez faire des commits régulièrement, utiliser un nom de dépôt Git représentatif et facile à identifier pour l'enseignant. Vous devez également faire en sorte que l'enseignant aie accès à votre code source au moment de la remise.

Évaluation basée sur :

1. **Respect des exigences fonctionnelles** (chaque module fonctionne comme prévu).
2. **Qualité du code** (organisation, clarté, commentaires, bonne pratique).
3. **Expérience utilisateur** (interface intuitive, animations, fluidité des interactions).
4. **Personnalisation** (possibilité de modifier l'apparence et la structure du Dashboard).
5. **Gamification** (implémentation du système d'XP, titres, niveaux et pénalités).
6. **Gestion des interactions** (ajout, suppression, modification et déplacement des modules).

Un bonus de maximum 10% pourra être accordé pour des fonctionnalités innovantes et pertinentes. Par exemple :

- Encryption des données.
- Importation/Exportation des données pour transférer sue un nouveau poste.
- Gestion avancée des thèmes (création de thèmes personnalisés par l'utilisateur).
- Gestion des to-do lists avec template. par exemple à chaque jour ajouter automatiquement des tâches selon un template configuré : Douche, brosser les dents, etc.
- Style global de l'application ou thème extrêmement personnalisable : Ex : thème pokémon, star wars, john wick, etc...
- Possibilité d'avoir un même module plusieurs fois affiché, mais avec des données différentes.
- Affichage multilingue.

Vous aurez accès à la grille d'évaluation sous peu.