IFOSUP
Institut de Formation Supérieure
Ville de Wavre

Nom & prénom de l'étudiant.e : .....

Section : BES Webdeveloper
Unité d'enseignement : Framework côté serveur (5XCOS)
Nom du/des chargé(s) de cours : Thibault Six
Année scolaire : 2026-2027

POUR ATTEINDRE LE SEUIL DE RÉUSSITE, L'ÉTUDIANT.E PROUVERA QU'IL/ELLE EST CAPABLE :

face au matériel et au logiciel adéquats et face à une structure informatique opérationnelle connectée à Internet, disposant des logiciels appropriés et de la documentation nécessaire et en respectant les normes et standards en vigueur, et au départ d'un nouveau cahier des charges ou du cahier des charges initial augmenté d'une nouvelle fonctionnalité ainsi que d'un framework et d'un modèle d'application proposés ou validés par le chargé de cours.

Acquis d'apprentissage	ACQUIS	NON ACQUIS
AA 1 : Déployer et exploiter le framework répondant aux consignes/objectifs figurant dans le cahier des charges.	☐	☐

AA 1.1 : Implémenter les fonctionnalités du cahier des charges.

☐ Niveau 1 : Moins de la moitié des fonctionnalités obligatoires (O1 à O12) sont implémentées ou fonctionnent. L'application ne permet pas de tenir l'inventaire des objets trouvés (liste, fiche, ajout).

☐ Niveau 2 : Au moins la moitié des fonctionnalités obligatoires sont présentes, mais des bugs majeurs ou des oublis critiques (formulaire qui n'enregistre pas, étiquettes perdues, filtre inopérant, restitution ou suppression absente) empêchent une utilisation complète.

☐ Niveau 3 : Toutes les fonctionnalités obligatoires O1 à O12 sont implémentées et fonctionnent : base et données du seeder, liste filtrée par lieu et par couleur, fiche d'un objet, page d'un lieu, ajout, modification, restitution, suppression, mise en forme. L'application s'utilise de bout en bout sans erreur.

☐ Niveau 4 : Toutes les fonctionnalités obligatoires fonctionnent sans défaut et au moins une fonctionnalité supplémentaire de l'énoncé est bien intégrée (même style que le reste, validée, présentée dans le rapport). Une supplémentaire ne compense jamais une obligatoire manquante.

AA 1.2 : Produire un code de qualité, fonctionnel et respectant les bonnes pratiques de programmation et les principes de l'orienté objet.

☐ Niveau 1 : Le code ne fonctionne pas ou est illisible : erreurs à l'exécution, logique entassée dans les vues ou les routes, noms incohérents, aucune convention du cours respectée.

☐ Niveau 2 : Le code fonctionne en partie. Lisibilité ou structure faibles (contrôleurs longs, validation recopiée dans chaque méthode, identifiants mi-anglais mi-français, code mort laissé en commentaire), mais les intentions restent discernables.

☐ Niveau 3 : Le code fonctionne, est lisible et suit les conventions Laravel du cours : identifiants en anglais, adresses et textes affichés en français, contrôleurs courts, validation dans une Form Request, attribut `#[Fillable]` sur les modèles, aucune requête dans les vues. La POO vue au cours est appliquée (classes, méthodes, encapsulation, enum avec sa méthode `label()`).

☐ Niveau 4 : Le code est exemplaire : validation complète et précise, composants Blade réutilisés partout où ils servent, logique placée au bon endroit (relations, `casts()`, méthodes de modèle et d'enum), aucune duplication, Pint propre (`vendor/bin/pint --test` sans remarque).

AA 1.3 : Gérer le contrôle de version du projet à l'aide de Git.

☐ Niveau 1 : Le dépôt GitHub est absent ou inaccessible au chargé de cours, ou Git n'est pas réellement utilisé : un ou deux commits contiennent tout le projet (« final », « Add files via upload »), aucun historique du développement.

☐ Niveau 2 : Git est utilisé, mais les commits sont irréguliers et peu parlants (« fix », « update », « wip »), les étapes sont mélangées, des fichiers qui n'ont rien à faire dans le dépôt y sont (`.env`, `vendor/`, `database.sqlite`) : l'historique se lit mal.

☐ Niveau 3 : Le dépôt GitHub est accessible au chargé de cours, les commits sont réguliers tout au long du projet et l'historique montre l'avancement du travail. Le `.gitignore` est respecté.

☐ Niveau 4 : Git est utilisé de manière exemplaire : messages de commit complets et descriptifs, branches pour les fonctionnalités supplémentaires ou tags aux étapes clés.

AA 1.4 : Rédiger un rapport technique structuré et pertinent.

☐ Niveau 1 : Le rapport n'est pas remis en PDF avec le nom de l'étudiant.e, ou il ne suit pas le canevas : sections manquantes, aucune illustration, vocabulaire technique absent ou erroné.

☐ Niveau 2 : Le rapport suit le canevas en partie : diagramme de classes absent ou ne respectant pas la norme UML, modèles et leurs relations (et contraintes si applicables : composition ou agrégation) non décrits, captures rares, difficultés à peine évoquées. Le vocabulaire technique est parfois imprécis.

Acquis d'apprentissage	ACQUIS	NON ACQUIS
☐ Niveau 3 : Le rapport est complet selon le canevas : diagramme de classes des modèles, description des modèles et de leurs relations, et des contraintes si applicables (composition ou agrégation), captures des pages, difficultés rencontrées et solutions, autocritique. Le vocabulaire technique est juste (MVC, migration, Eloquent, Form Request, Blade). ☐ Niveau 4 : Le rapport est d'une qualité professionnelle : choix techniques justifiés, difficultés analysées avec leur solution, pistes d'amélioration réalistes, mise en page soignée, contribution de l'IA bien documentée (ce qu'elle a produit, ce qui a été vérifié et corrigé). **AA 1.5 : Rendre compte de son usage de l'IA dans le carnet métacognitif.** ☐ Niveau 1 : Le carnet n'est pas remis en PDF avec le nom de l'étudiant.e, ou il est vide ou ne dit rien de l'usage de l'IA pendant le projet. ☐ Niveau 2 : Le carnet est rempli de façon générique (« j'ai utilisé l'IA pour m'aider ») : aucun exemple, aucun prompt, rien sur ce qui a été vérifié. On ne sait pas ce qui vient de l'étudiant.e. ☐ Niveau 3 : Le carnet est honnête et concret : exemples de prompts et de réponses, vérifications décrites (documentation Laravel, essai dans le navigateur, `php artisan route:list`), au moins une ligne de code proposée par l'IA expliquée avec ses propres mots. ☐ Niveau 4 : Le carnet porte un regard critique argumenté : erreurs de l'IA identifiées et corrigées (code d'une ancienne version, convention non respectée, faille), limites de l'outil discutées, ce qui revient à l'étudiant.e clairement distingué de ce qui vient de l'IA.		

Tous les AA sont « acquis »	Au moins un AA non acquis (en 1re session)	Au moins un AA non acquis (en 2e session)
☐ **RÉUSSITE** **→ Voir degré de maîtrise**	☐ **AJOURNEMENT** **→ Voir justification ajournement**	☐ **REFUS** **→ Voir justification refus**

DEGRÉ DE MAÎTRISE (% ≥ 50) - 0 = Acquis, 1 = Bien, 2 = TB, 3 = TTB, 4 = Excellent et 5 = Parfait

Critères	Indicateurs	Niveau
N°1 **Qualité de l'interface et ergonomie** Tailwind CSS, responsive, composants, cohérence visuelle	☐ **0/5 (0/10) : Interface absente ou inutilisable.** Pages sans mise en forme, Tailwind CSS non installé ou non utilisé, rien ne s'affiche correctement à 375 px de large. ☐ **1/5 (2/10) : Interface rudimentaire.** Quelques classes Tailwind éparses, pas de layout commun ni de navigation, liste illisible sur mobile. ☐ **2/5 (4/10) : Interface fonctionnelle mais peu soignée.** Le layout `<x-layouts.app>` existe, mais couleurs et espacements changent d'une page à l'autre, badges ou états `hover`/`focus` manquent, le mobile est en partie cassé. ☐ **3/5 (6/10) : Interface claire.** Layout avec navigation, trois composants réutilisés (`<x-button>`, `<x-badge>`, `<x-card>`), badges colorés selon l'enum, états `hover`/`focus` présents, liste lisible à 375 px avec de petits défauts. ☐ **4/5 (8/10) : Interface soignée et cohérente.** Une palette tenue sur toutes les pages, hiérarchie visuelle nette (titres, cartes, formulaires), messages flash et erreurs bien visibles, application entièrement responsive. ☐ **5/5 (10/10) : Interface exemplaire.** Ergonomie pensée pour l'agent du bureau (filtres simples, actions évidentes, confirmation claire avant suppression), composants homogènes réutilisés partout, labels, contrastes et focus visibles, rendu impeccable du mobile au grand écran.	/10
N°2 **Validation, robustesse et sécurité** règles, messages en français, `old()`, flash, CSRF, échappement, cas limites	☐ **0/5 (0/10) : Aucune validation.** Les formulaires acceptent n'importe quoi ou plantent (erreur 500, `MassAssignmentException`), `@csrf` absent, données affichées sans échappement (`{!! !!}`). ☐ **1/5 (2/10) : Validation minimale.** Quelques `required` seulement, messages en anglais ou absents, saisie perdue après un refus, date dans le futur acceptée, objet restituable deux fois. ☐ **2/5 (4/10) : Validation incomplète.** Une partie des règles imposées manque (longueur du nom, liste des couleurs, `exists`, date pas dans le futur), erreurs affichées en bloc plutôt que sous chaque champ, `old()` sur une partie des champs, flash absent après certaines actions. ☐ **3/5 (6/10) : Validation correcte.** Toutes les règles imposées dans `ItemRequest`, erreurs en français sous chaque champ (Laravel-Lang, `attributes()`), `old()` sur tous les champs, message flash après chaque action, `@csrf` partout, `#[Fillable]` sur les modèles. ☐ **4/5 (8/10) : Application robuste.** Cases des étiquettes qui respectent `old()` après un refus, restitution double refusée avec un message clair, suppression en cascade vérifiée (les lignes de `item_tag` disparaissent), confirmation avant suppression, aucune sortie non échappée. ☐ **5/5 (10/10) : Robustesse exemplaire.** Cas limites prévus et vérifiés (filtre avec une valeur inconnue, objet inexistant en 404, champ trop long, étiquette décochée), aucun chemin ne mène à une erreur 500, choix de validation et de sécurité expliqués dans le rapport.	/10
N°3 **Architecture et respect des conventions Laravel** routes nommées, liaison de modèle, Form Request, relations Eloquent, seeders et fabriques, enums	☐ **0/5 (0/10) : Architecture non conforme à Laravel.** Requêtes SQL dans les vues ou les routes, pas de contrôleur ni de modèle Eloquent, migrations absentes ou fausses. ☐ **1/5 (2/10) : Application difficile à maintenir.** Routes en closure, contrôleurs fourre-tout, relations Eloquent absentes (identifiants manipulés à la main), pas de seeder, logique dans les vues. ☐ **2/5 (4/10) : MVC rudimentaire.** Contrôleurs et modèles présents, mais routes non nommées ou adresses en anglais, pas de liaison de modèle (`find($id)` partout), relations incomplètes, données aléatoires qui changent la démo à chaque `migrate:fresh --seed`.	/10

Critères	Indicateurs	Niveau
	☐ **3/5 (6/10) : Bonne structure.** Tableau des routes respecté (adresses en français, noms en anglais, `route()` dans les vues), liaison de modèle `{item}`, relations `hasMany`/`belongsTo`/`belongsToMany`, `constrained()->cascadeOnDelete()`, enum `PlaceKind` et `casts()`, fabriques et seeder fixe. ☐ **4/5 (8/10) : Architecture bien pensée.** Contrôleurs courts qui délèguent (Form Request partagée, relations, `sync()`), relations chargées sans requêtes en boucle (`with()`), seeder fidèle à l'énoncé (4 lieux, 8 étiquettes, 15 objets dont 3 restitués), vues propres. ☐ **5/5 (10/10) : Architecture exemplaire.** Chaque responsabilité à sa place (filtres en scopes ou méthodes de modèle, enums avec libellé et couleur, composants Blade modulaires), aucune duplication, application prête à évoluer (une table ou une page de plus sans tout réécrire).	
N°4 **Qualité de l'analyse et de la réflexion dans le rapport** diagramme, choix expliqués, difficultés, autocritique	☐ **0/5 (0/10) : Aucune analyse.** Le rapport décrit les écrans sans rien expliquer, les illustrations sont absentes ou inutiles, le contenu est générique et pourrait valoir pour n'importe quel projet. ☐ **1/5 (2/10) : Rapport superficiel.** Liste des fonctionnalités sans analyse, diagramme de classes absent, captures rares ou illisibles. ☐ **2/5 (4/10) : Analyse limitée.** Choix justifiés en une phrase, diagramme de classes présent mais incomplet ou faux (relations, cardinalités), difficultés citées sans leur solution. ☐ **3/5 (6/10) : Analyse correcte.** Diagramme de classes juste (`Place`, `Item`, `Tag`, relations et cardinalités), choix techniques expliqués (enum, Form Request partagée, table de liaison), difficultés décrites avec leur solution. ☐ **4/5 (8/10) : Analyse approfondie.** Choix comparés à d'autres possibles, difficultés analysées (cause, piste suivie, solution retenue), autocritique honnête de ce qui reste perfectible, illustrations nombreuses et utiles (extraits de code commentés, captures annotées). ☐ **5/5 (10/10) : Rapport de référence.** Réflexion personnelle et argumentée sur le projet et sur sa façon d'apprendre, autocritique lucide, pistes d'amélioration concrètes, diagramme impeccable, rédaction claire et professionnelle.	/10
N°5 **Fonctionnalités supplémentaires** S1 à S16 de l'énoncé : effort en étoiles, style du site, validation, présentation dans le rapport	☐ **0/5 (0/10) : Aucune supplémentaire.** Rien au-delà des fonctionnalités obligatoires O1 à O13, ou les tentatives ne fonctionnent pas. ☐ **1/5 (2/10) : Supplémentaire esquissée.** Une supplémentaire tentée mais incomplète, qui casse une obligatoire ou qui sort du style du site. ☐ **2/5 (4/10) : Supplémentaires simples.** Une ou deux supplémentaires ★ (pagination, recherche, tri, dates en français, mode sombre…) qui fonctionnent, dans le style du site. ☐ **3/5 (6/10) : Supplémentaires bien faites.** Trois supplémentaires ou plus qui fonctionnent, dans le style du site, validées quand elles reçoivent une saisie et présentées dans le rapport. ☐ **4/5 (8/10) : Supplémentaires ambitieuses.** Le niveau 3, avec au moins une supplémentaire ★★ ou ★★★ complète et bien faite (galerie, déclarations de perte, statistiques, tests et CI, mise en ligne, deux langues), ou cinq supplémentaires ★ ou plus toutes bien faites. ☐ **5/5 (10/10) : Supplémentaires de référence.** Plusieurs supplémentaires ★★ ou ★★★ complètes et robustes, sans régression des obligatoires, chacune justifiée dans le rapport (pourquoi, page de la documentation lue).	/10

CALCUL DE LA NOTE FINALE	
Seuil de réussite (tous AA acquis) :	/ 50
Total des critères de maîtrise :	/ 50
NOTE FINALE :	/ 100

Date : **Signature du/des chargé(s) de cours :**