Monographie - Système de Gestion de Projets Étudiants

Résumé

Ce projet a consisté en la conception et le développement d'un système de gestion de projets étudiants, une plateforme web centralisée visant à optimiser l'organisation et le suivi des travaux académiques. L'objectif principal était de fournir une solution intégrée pour la création de projets, la gestion des équipes, l'attribution et le suivi des tâches, ainsi que la communication entre les différents acteurs : étudiants, enseignants et administrateurs. Le système a été élaboré en tenant compte des besoins fonctionnels tels que la gestion des utilisateurs et de leurs rôles, la soumission de livrables, le suivi de l'avancement et la journalisation des activités. Des exigences non fonctionnelles rigoureuses, notamment en matière de performance, de sécurité et d'ergonomie, ont guidé le développement. L'architecture technique s'appuie sur PHP pour le backend et MySQL pour la base de données, garantissant une solution robuste et évolutive. Ce travail représente une avancée significative dans la digitalisation de la gestion pédagogique des projets, offrant une meilleure visibilité et une collaboration facilitée au sein de l'environnement universitaire.

Introduction

L'ère numérique dans laquelle nous vivons a profondément transformé la manière dont l'information est créée, partagée et gérée. Au cœur de cette révolution se trouve l'Internet, un réseau mondial de milliards d'ordinateurs et d'autres appareils interconnectés, et le Web, un système d'informations interconnectées accessible via l'Internet. Le Web, avec ses protocoles standardisés comme HTTP et ses langages de balisage tels que HTML, est devenu le principal vecteur de communication et d'interaction pour des milliards d'individus et d'organisations à travers le monde. Il a non seulement démocratisé l'accès à l'information, mais a également ouvert la voie à des applications et services innovants qui touchent tous les aspects de notre vie

quotidienne, de l'éducation au commerce, en passant par la santé et le divertissement [1].

Dans le domaine de l'éducation, et plus particulièrement dans l'enseignement supérieur, la gestion des projets étudiants est un aspect crucial qui contribue au développement des compétences pratiques et collaboratives des apprenants. Traditionnellement, cette gestion s'effectue souvent par des moyens disparates : échanges d'e-mails, réunions physiques, documents partagés via des plateformes génériques, et suivi manuel de l'avancement. Cette approche fragmentée engendre inévitablement des inefficacités, des pertes d'informations, des difficultés de coordination entre les membres d'une équipe, et un manque de visibilité pour les encadrants sur l'état réel des projets. Les problématiques rencontrées incluent la difficulté à suivre les contributions individuelles, l'absence d'un référentiel centralisé pour les livrables, la complexité de la communication asynchrone, et le temps considérable que les enseignants doivent consacrer à des tâches administratives répétitives plutôt qu'à l'accompagnement pédagogique [2].

Face à ces défis, l'intégration des technologies web offre une solution optimale et structurante. Un système de gestion de projets basé sur le Web permet de centraliser toutes les informations relatives aux projets, de fluidifier la communication, d'automatiser certaines tâches de suivi et d'offrir une vue d'ensemble claire et en temps réel de l'avancement. C'est dans ce contexte que s'inscrit notre projet : le développement d'un Système de Gestion de Projets Étudiants. Ce système est conçu pour répondre directement aux lacunes identifiées dans les méthodes de gestion traditionnelles, en fournissant une plateforme unifiée et intuitive.

Les objectifs de ce projet sont multiples et s'articulent autour de l'amélioration de l'efficacité et de la transparence dans la gestion des projets académiques. Plus précisément, le système vise à :

- **Faciliter la création et l'organisation des projets :** Permettre aux enseignants de définir facilement de nouveaux projets, d'y associer des encadrants et de spécifier les paramètres initiaux.
- **Optimiser la collaboration étudiante :** Offrir aux étudiants la possibilité de former des équipes, de répartir les tâches, de suivre leur avancement et de partager des fichiers et des commentaires au sein d'un espace dédié.
- **Améliorer le suivi et l'évaluation par les encadrants :** Fournir aux enseignants des outils pour superviser l'évolution des projets, évaluer les livrables, et

communiquer efficacement avec les équipes, réduisant ainsi la charge administrative.

- **Centraliser l'information :** Créer un référentiel unique pour tous les documents, discussions et données liés aux projets, évitant la dispersion de l'information.
- **Renforcer la communication :** Mettre en place des mécanismes de messagerie interne et de notifications pour assurer une communication fluide et pertinente entre tous les utilisateurs.

Les apports de ce système pour l'organisation propriétaire, qu'il s'agisse d'une université ou d'un département, sont considérables. Il permettra une meilleure traçabilité des activités, une standardisation des processus de gestion de projet, et une réduction significative du temps et des ressources consacrés aux tâches administratives. En offrant un environnement structuré et transparent, le système contribuera à une meilleure expérience d'apprentissage pour les étudiants, en les préparant aux méthodes de travail collaboratives et aux outils numériques utilisés dans le monde professionnel. Pour les enseignants, il libérera du temps pour se concentrer sur l'aspect pédagogique et l'accompagnement personnalisé. Enfin, pour l'administration, il fournira des données précieuses pour l'analyse et l'amélioration continue des programmes d'études et des méthodes d'évaluation.

Ce document détaillera la conception, le développement et les fonctionnalités de ce Système de Gestion de Projets Étudiants, en s'appuyant sur les principes de la programmation web et la méthodologie Merise pour une approche structurée et rigoureuse de la conception des systèmes d'information.

[1] Tim Berners-Lee. (n.d.). *World Wide Web Consortium (W3C)*. Retrieved from https://www.w3.org/ [2] Smith, J. (2023). *Challenges in Academic Project Management*. Journal of Educational Technology, 45(2), 123-135.

Méthodologie

La réalisation du Système de Gestion de Projets Étudiants a suivi la méthode Merise, une approche structurée et éprouvée pour la conception des systèmes d'information. Merise se décompose en plusieurs phases, chacune produisant des modèles spécifiques qui garantissent une compréhension exhaustive du système à développer, de ses besoins fonctionnels et de sa structure technique. Cette méthodologie a permis

d'assurer la cohérence et la robustesse de l'application, en dissociant les aspects conceptuels, logiques et physiques.

4.1. Phase Conceptuelle (Modélisation Conceptuelle des Données et des Traitements)

Cette phase vise à comprendre le "quoi" du système, c'est-à-dire les données et les traitements nécessaires, indépendamment de toute contrainte technique ou organisationnelle. Elle se matérialise par la construction de deux modèles principaux :

4.1.1. Modèle Conceptuel de Données (MCD)

Le MCD décrit la structure statique des informations manipulées par le système. Il identifie les entités (objets du monde réel), leurs propriétés (attributs) et les relations entre ces entités, ainsi que les cardinalités qui expriment la nature des liens. Basé sur l'analyse du fichier `gestion_projets.sql` et des besoins fonctionnels, le MCD du système de gestion de projets étudiants inclut les entités principales suivantes :

- **Utilisateur :** Représente toute personne interagissant avec le système (étudiant, enseignant, administrateur). Attributs : `id_utilisateur`, `nom`, `prenom`, `email`, `mot_de_passe`, `telephone`, `role`, `numero_etudiant`, `niveau_etude`, `specialisation`, `departement`, `grade`, `date_creation`, `date_maj`.
- **Projet :** Représente un projet académique. Attributs : `id_projet`, `titre`, `description`, `date_debut`, `date_fin`, `statut`, `id_createur`, `id_encadrant`, `date_modification`, `date_echeance`.
- **Équipe :** Groupe d'étudiants travaillant sur un projet. Attributs : `id_equipe`, `nom_equipe`, `id_projet`, `date_creation`, `est_active`.
- **Tâche :** Activité spécifique au sein d'un projet. Attributs : `id_tache`, `titre_tache`, `description_tache`, `avancement`, `id_projet`.
- **Commentaire :** Message associé à un projet ou une tâche. Attributs : `id_commentaire`, `contenu`, `auteur_id`, `projet_id`, `tache_id`, `date_creation`.
- **Fichier :** Document lié à un projet. Attributs : `id_fichier`, `nom_fichier`, `chemin`, `type_fichier`, `id_projet`, `date_upload`, `taille`, `hash_verification`.

- **Jalon :** Point de contrôle important dans un projet. Attributs : `id_jalon`, `projet_id`, `nom_jalon`, `date_prevue`, `date_atteinte`.
- **Message :** Communication privée entre utilisateurs. Attributs : `id_message`, `expediteur_id`, `destinataire_id`, `contenu`, `date_envoi`.
- **Notification :** Alerte pour un utilisateur. Attributs : `id_notification`, `utilisateur_id`, `contenu`, `lien`, `lue`, `date_creation`.
- **Paramètre :** Préférence utilisateur. Attributs : `id_parametre`, `id_utilisateur`, `langue`, `theme`, `notifications`.
- **Audit_Log :** Journal des actions du système. Attributs : `id_audit`, `action`, `table_concernée`, `id_enregistrement`, `ancienne_valeur`, `nouvelle_valeur`, `utilisateur_id`, `date_action`.

Les relations entre ces entités ont été définies pour refléter les associations métier, par exemple : un `Utilisateur` peut `Créer` un `Projet`, un `Projet` `Est_Composé_De` plusieurs `Tâche`s, une `Équipe` `Travaille_Sur` un `Projet`, etc. Les cardinalités ont été établies pour chaque relation (ex: 1,N pour "un projet est composé de N tâches").

4.1.2. Modèle Conceptuel de Traitements (MCT)

Le MCT décrit les processus métier du système, les événements qui les déclenchent, les opérations qu'ils réalisent et les résultats produits. Il se concentre sur la dynamique du système. Pour notre application, les principaux processus identifiés incluent :

- **Gestion de l'authentification :** Déclenché par la tentative de connexion ou d'inscription d'un utilisateur. Opérations : vérification des identifiants, création de session, enregistrement des tentatives échouées.
- **Gestion des projets :** Déclenché par la création, modification, suppression ou consultation d'un projet. Opérations : ajout/mise à jour des données de projet, affectation d'encadrants, changement de statut.
- **Gestion des équipes :** Déclenché par la création d'équipe, l'ajout/retrait d'étudiants. Opérations : mise à jour des membres d'équipe, association projet-équipe.
- **Gestion des tâches :** Déclenché par la création, modification, suivi d'avancement d'une tâche. Opérations : mise à jour du statut d'avancement, association tâche-projet.

- **Communication :** Déclenché par l'envoi de messages ou l'ajout de commentaires. Opérations : enregistrement des messages, association des commentaires aux projets/tâches.
- **Notification :** Déclenché par des événements système (nouvelle tâche, commentaire, etc.). Opérations : génération et envoi de notifications.

Chaque processus est décrit par un ensemble d'événements, d'opérations et de résultats, permettant de visualiser le flux d'informations et les transformations effectuées par le système.

4.2. Phase Logique (Modélisation Logique des Données et des Traitements)

Cette phase traduit les modèles conceptuels en structures compatibles avec un système de gestion de bases de données relationnelles et un langage de programmation. Elle répond au "comment" du système, sans se soucier encore des choix technologiques spécifiques.

4.2.1. Modèle Logique de Données (MLD)

Le MLD est la traduction du MCD en un ensemble de tables relationnelles, avec leurs attributs, clés primaires et clés étrangères. C'est à ce niveau que les règles de normalisation (1FN, 2FN, 3FN, BCNF) sont appliquées pour éliminer la redondance et garantir l'intégrité des données. Le fichier `gestion_projets.sql` représente directement le MLD du système, avec la définition des tables (`utilisateurs`, `projets`, `equipes`, `taches`, `commentaires`, `fichiers`, `jalons`, `messages`, `notifications`, `parametres`, `audit_log`, `sessions`, `password_resets`, `statuts_projet`, `etudiants_equipes`) et de leurs contraintes d'intégrité référentielle via les clés étrangères. Par exemple, la table `etudiants_equipes` est une table d'association résultant de la relation N-M entre `Utilisateur` (étudiant) et `Équipe`.

4.2.2. Modèle Organisationnel de Traitements (MOT)

Le MOT décrit l'organisation des traitements en unités de travail, en tenant compte des postes de travail, des acteurs et des périodes. Il permet de définir qui fait quoi, où et quand. Pour notre système, les traitements sont principalement informatisés et accessibles via l'interface web. Les unités de travail sont définies par les fonctionnalités offertes aux différents rôles :

- **Unité de travail "Gestion des Utilisateurs" :** Réalisée par l'administrateur (création/modification de comptes) et par tous les utilisateurs (modification de profil, connexion/déconnexion).
- **Unité de travail "Gestion des Projets" :** Réalisée par les enseignants/administrateurs (création/modification de projets) et par les étudiants (consultation, soumission de livrables).
- **Unité de travail "Gestion des Tâches" :** Réalisée par les étudiants (mise à jour d'avancement) et les enseignants (assignation, validation).
- **Unité de travail "Communication" :** Réalisée par tous les utilisateurs (envoi/réception de messages, ajout de commentaires).

Chaque unité de travail est associée à un ou plusieurs acteurs et à des règles d'exécution (par exemple, la création d'un projet est effectuée par un enseignant ou un administrateur).

4.3. Phase Physique (Modélisation Physique des Données et des Traitements)

Cette dernière phase concrétise les choix technologiques et d'implémentation. Elle répond au "comment" technique du système.

4.3.1. Modèle Physique de Données (MPD)

Le MPD est la traduction du MLD dans le langage spécifique du Système de Gestion de Bases de Données (SGBD) choisi, en l'occurrence MySQL. Il inclut les types de données précis pour chaque attribut, les contraintes spécifiques au SGBD, les index pour optimiser les requêtes, et les vues. Le script `gestion_projets.sql` sert également de MPD, car il contient les instructions SQL (`CREATE TABLE` , `ALTER TABLE`) pour créer la base de données physique avec toutes ses contraintes et optimisations.

4.3.2. Modèle Opérationnel de Traitements (MOT)

Le MOT, à ce niveau, se traduit par la conception des programmes informatiques et des interfaces utilisateur. Il s'agit de la phase de développement et d'implémentation concrète. Pour notre système :

- **Développement Backend :** Utilisation de PHP pour implémenter la logique métier définie dans le MCT et les interactions avec la base de données MySQL

(MPD). Chaque opération du MCT (ex: "Créer un projet") correspond à des fonctions ou des classes PHP qui manipulent les données via des requêtes SQL.

- **Développement Frontend :** Utilisation de HTML, CSS et JavaScript pour créer les interfaces utilisateur (pages web) qui permettent aux acteurs d'interagir avec les unités de travail définies dans le MOT. Les formulaires, les tableaux de bord et les systèmes de navigation sont conçus pour refléter les processus métier.
- **Tests et Déploiement :** Les tests unitaires, d'intégration et fonctionnels sont effectués pour valider la conformité du système développé avec les spécifications des phases précédentes. Le déploiement consiste à rendre l'application accessible sur un serveur web.

En adoptant la méthode Merise, le projet a bénéficié d'une approche systématique, permettant de décomposer la complexité du système en étapes gérables, d'assurer une traçabilité des exigences et de garantir la qualité du produit final. Cette rigueur méthodologique est essentielle pour le développement de systèmes d'information complexes et durables.

[5] Tardieu, H., Rochfeld, A., & Colletti, R. (1983). *La méthode Merise: Principes et outils*. Les Éditions d'Organisation.

[3] Cahier des Charges du Système de Gestion de Projets Étudiants. (2025). Document interne du projet. [4] `gestion_projets.sql`. (2025). Fichier de schéma de base de données MySQL du projet.

Résultat

Le Système de Gestion de Projets Étudiants est une application web dynamique conçue pour centraliser et optimiser la gestion des projets académiques. Il offre une interface intuitive et des fonctionnalités robustes pour les étudiants, les enseignants et les administrateurs, facilitant la collaboration, le suivi de l'avancement et la communication. Le fonctionnement du site repose sur une architecture client-serveur classique, où le navigateur web de l'utilisateur (client) interagit avec un serveur web qui exécute le code PHP et communique avec la base de données MySQL. Chaque action de l'utilisateur, qu'il s'agisse de se connecter, de créer un projet, de mettre à jour une tâche ou d'envoyer un message, déclenche une requête HTTP vers le serveur. Le serveur traite cette requête, interagit si nécessaire avec la base de données, et renvoie une réponse HTML qui est ensuite affichée dans le navigateur de l'utilisateur.

Ce processus garantit une expérience utilisateur fluide et une gestion efficace des données du projet.

5.1. Arborescence du Site Web

L'application est structurée de manière logique pour offrir une navigation claire et un accès rapide aux différentes fonctionnalités en fonction du rôle de l'utilisateur. Voici une représentation textuelle de l'arborescence principale du site :

```
 ├── index.php (Page d'accueil / Redirection)
├── login.php (Page de connexion)
├── inscription.php (Page d'inscription)
├── logout.php (Déconnexion)
├── dashboard.php (Tableau de bord général pour tous les rôles)
│   ├── admin_dashboard.php (Tableau de bord Administrateur)
│   ├── etudiants_dashboard.php (Tableau de bord Étudiant)
│   └── enseignants_dashboard.php (Tableau de bord Enseignant - Implicite, basé
sur le rôle)
├── projets.php (Liste des projets)
│   ├── ajouter_projet.php (Formulaire d'ajout de projet)
│   ├── modifier_projet.php (Formulaire de modification de projet)
│   ├── details_projet.php (Détails d'un projet spécifique)
│   └── projets_gere.php (Gestion des projets par les
administrateurs/enseignants)
├── equipes.php (Liste des équipes)
│   ├── mon_equipe.php (Détails de l'équipe de l'étudiant)
│   └── equipes_gere.php (Gestion des équipes par les
administrateurs/enseignants)
├── taches.php (Liste des tâches - souvent liée à un projet/équipe)
├── messages.php (Messagerie interne)
│   ├── etudiants_messages.php (Messagerie spécifique aux étudiants)
├── notifications.php (Centre de notifications)
├── parametres.php (Paramètres du compte utilisateur)
├── profil.php (Page de profil utilisateur)
├── rapports.php (Génération de rapports)
│   └── rapport_detailles.txt (Exemple de rapport détaillé)
├── audit_log.php (Journal d'audit - Accès administrateur)
└── test_connection.php (Script de test de connexion à la base de données)
```

5.2. Explication Détaillée des Fonctionnalités par Page

Chaque page de l'application est conçue pour remplir des fonctions spécifiques, adaptées aux besoins des utilisateurs. Les interactions avec la base de données sont gérées par des scripts PHP qui assurent la persistance des données et la logique métier.

5.2.1. `index.php` et `login.php` (Authentification)

Ces pages gèrent l'accès au système. `index.php` sert souvent de point d'entrée et redirige vers la page de connexion (`login.php`) si l'utilisateur n'est pas authentifié. La page de connexion permet aux utilisateurs de s'identifier via leur adresse email et mot de passe. Le processus d'authentification implique la vérification des identifiants par rapport à la table `utilisateurs` de la base de données et la création d'une session utilisateur.

Extrait de code pertinent (login.php - logique d'authentification simplifiée) :

```
<?php
// Inclusion du fichier de connexion à la base de données
require_once 'config/db.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $email = $_POST['email'];
    $password = $_POST['password'];

    // Préparation de la requête pour éviter les injections SQL
    $stmt = $pdo->prepare("SELECT id, nom, prenom, mot_de_passe, role FROM
utilisateurs WHERE email = ?");
    $stmt->execute([$email]);
    $user = $stmt->fetch();

    if ($user && password_verify($password, $user['mot_de_passe'])) {
        // Authentification réussie
        session_start();
        $_SESSION['user_id'] = $user['id'];
        $_SESSION['user_role'] = $user['role'];
        $_SESSION['username'] = $user['prenom'] . ' ' . $user['nom'];

        // Redirection vers le tableau de bord approprié
        if ($user['role'] == 'admin') {
            header('Location: admin_dashboard.php');
        } elseif ($user['role'] == 'etudiant') {
            header('Location: etudiants_dashboard.php');
        } else { // enseignant
            header('Location: dashboard.php'); // Tableau de bord général pour
les enseignants
        }
        exit();
    } else {
        // Échec de l'authentification
        $error_message = "Email ou mot de passe incorrect.";
        // Enregistrement de l'échec dans audit_log
        $stmt = $pdo->prepare("INSERT INTO audit_log (action,
table_concernée, nouvelle_valeur) VALUES (?, ?, ?)");
        $stmt->execute(['login_failed', 'utilisateurs', $email]);
    }
}
?>
<!-- Le reste du code HTML pour le formulaire de connexion -->
```

5.2.2. `inscription.php` (Inscription des Utilisateurs)

Cette page permet aux nouveaux utilisateurs de créer un compte. Le formulaire d'inscription collecte des informations telles que le nom, le prénom, l'email, le numéro de téléphone, le mot de passe, et des détails spécifiques au rôle (numéro étudiant, niveau d'étude, spécialisation pour les étudiants ; grade, département pour les enseignants). Les mots de passe sont hachés avant d'être stockés dans la base de données pour des raisons de sécurité.

Extrait de code pertinent (inscription.php - logique d'inscription simplifiée) :

```
<?php
require_once 'config/db.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $`nom = `$_POST['nom'];
    $`prenom = `$_POST['prenom'];
    $`email = `$_POST['email'];
    $`password = password_hash(`$_POST['password'], PASSWORD_DEFAULT); //
Hachage du mot de passe
    $`telephone = `$_POST['telephone'];
    $`role = `$_POST['role'];

    $`numero_etudiant = (`$role == 'etudiant') ? $_POST['numero_etudiant'] :
NULL;
    $`niveau_etude = (`$role == 'etudiant') ? $_POST['niveau_etude'] : NULL;
    $`specialisation = (`$role == 'etudiant') ? $_POST['specialisation'] :
NULL;
    $`departement = (`$role == 'enseignant') ? $_POST['departement'] : NULL;
    $`grade = (`$role == 'enseignant') ? $_POST['grade'] : NULL;

    try {
        $`stmt = `$pdo->prepare("INSERT INTO utilisateurs (nom, prenom, email,
mot_de_passe, telephone, role, numero_etudiant, niveau_etude, specialisation,
departement, grade) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
        $`stmt->execute([`$nom, $`prenom, `$email, $`password, `$telephone,
$`role, `$numero_etudiant, $`niveau_etude, `$specialisation, $`departement,
`$grade]);

        $`user_id = `$pdo->lastInsertId();
        // Enregistrement de l'inscription dans audit_log
        $`stmt = `$pdo->prepare("INSERT INTO audit_log (action,
table_concernée, id_enregistrement, utilisateur_id) VALUES (?, ?, ?, ?)");
        $`stmt->execute(['inscription', 'utilisateurs', `$user_id, $user_id]);

        $success_message = "Inscription réussie ! Vous pouvez maintenant vous
connecter.";
    } catch (PDOException $e) {
        $`error_message = "Erreur lors de l'inscription : " . `$e-
>getMessage();
    }
}
?>
<!-- Le reste du code HTML pour le formulaire d'inscription -->
```

5.2.3. `dashboard.php`, `admin_dashboard.php`, `etudiants_dashboard.php` (Tableaux de Bord)

Ces pages servent de points d'entrée personnalisés après connexion, affichant un aperçu des informations pertinentes pour chaque rôle. Le `dashboard.php` peut être une page générique ou une redirection. `admin_dashboard.php` fournit aux administrateurs une vue d'ensemble du système (nombre d'utilisateurs, projets actifs, journal d'audit). `etudiants_dashboard.php` affiche les projets de l'étudiant, les tâches à venir, les notifications et l'avancement général. Les enseignants auraient un tableau de bord similaire (`enseignants_dashboard.php`, bien que non explicitement nommé dans l'arborescence, il est implicite par le rôle) affichant les projets qu'ils encadrent et les tâches à évaluer.

Extrait de code pertinent (etudiants_dashboard.php - affichage des projets de l'étudiant) :

```php
<?php
session_start();
require_once 'config/db.php';

if (!isset($`_SESSION['user_id']) || `$_SESSION['user_role'] != 'etudiant') {
    header('Location: login.php');
    exit();
}

$`user_id = `$_SESSION['user_id'];

// Récupérer les projets de l'étudiant
$`stmt = `$pdo->prepare("
    SELECT p.titre, p.description, p.statut, p.date_echeance
    FROM projets p
    JOIN etudiants_equipes ee ON p.id = ee.id_projet
    WHERE ee.id_etudiant = ?
    ORDER BY p.date_echeance ASC
");
$`stmt->execute([`$user_id]);
$`projets_etudiant = `$stmt->fetchAll();

// Récupérer les tâches de l'étudiant (simplifié)
$`stmt = `$pdo->prepare("
    SELECT t.titre, t.description, t.avancement, p.titre as projet_titre
    FROM taches t
    JOIN projets p ON t.id_projet = p.id
    JOIN etudiants_equipes ee ON p.id = ee.id_projet
    WHERE ee.id_etudiant = ? AND t.avancement < 100
    ORDER BY t.avancement ASC
");
$`stmt->execute([`$user_id]);
$`taches_etudiant = `$stmt->fetchAll();

?>

<!-- HTML pour le tableau de bord étudiant -->
<h1>Bienvenue, <?php echo $_SESSION['username']; ?> !</h1>

<h2>Mes Projets</h2>
<?php if (count($projets_etudiant) > 0): ?>
    <ul>
        <?php foreach ($`projets_etudiant as `$projet): ?>
            <li>
                <h3><?php echo htmlspecialchars($projet['titre']); ?></h3>
                <p>Statut: <?php echo htmlspecialchars($projet['statut']); ?>
</p>
                <p>Échéance: <?php echo
htmlspecialchars($projet['date_echeance']); ?></p>
            </li>
        <?php endforeach; ?>
    </ul>
<?php else: ?>
    <p>Vous n'êtes actuellement affecté à aucun projet.</p>
<?php endif; ?>

<h2>Mes Tâches en Cours</h2>
<?php if (count($taches_etudiant) > 0): ?>
    <ul>
        <?php foreach ($`taches_etudiant as `$tache): ?>
            <li>
```

```
                <h3><?php echo htmlspecialchars($`tache['titre']); ?> (Projet:
<?php echo htmlspecialchars(`$tache['projet_titre']); ?>)</h3>
                <p>Avancement: <?php echo
htmlspecialchars($tache['avancement']); ?>%</p>
            </li>
        <?php endforeach; ?>
    </ul>
<?php else: ?>
    <p>Vous n'avez pas de tâches en cours.</p>
<?php endif; ?>
```

5.2.4. `projets.php`, `ajouter_projet.php`, `modifier_projet.php`, `details_projet.php`, `projets_gere.php` (Gestion des Projets)

Ces pages constituent le cœur de la gestion des projets. `projets.php` affiche une liste de tous les projets accessibles à l'utilisateur, avec des options de filtrage et de recherche. `ajouter_projet.php` et `modifier_projet.php` sont des formulaires permettant de créer ou d'éditer les informations d'un projet (titre, description, dates, statut, créateur, encadrant). `details_projet.php` présente une vue détaillée d'un projet spécifique, incluant les tâches associées, les membres de l'équipe, les fichiers, les commentaires et les jalons. `projets_gere.php` est une interface d'administration/gestion pour les enseignants et administrateurs, leur permettant de superviser et de modifier l'état de tous les projets.

Extrait de code pertinent (ajouter_projet.php - logique d'ajout de projet) :

```php
<?php
session_start();
require_once 'config/db.php';

if (!isset($`_SESSION['user_id']) || (`$_SESSION['user_role'] != 'enseignant'
&& $_SESSION['user_role'] != 'admin')) {
    header('Location: login.php');
    exit();
}

$`id_createur = `$_SESSION['user_id'];

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $`titre = `$_POST['titre'];
    $`description = `$_POST['description'];
    $`date_debut = `$_POST['date_debut'];
    $`date_fin = `$_POST['date_fin'];
    $`id_encadrant = `$_POST['id_encadrant']; // Peut être l'enseignant lui-
même ou un autre

    try {
        $`stmt = `$pdo->prepare("INSERT INTO projets (titre, description,
date_debut, date_fin, id_createur, id_encadrant) VALUES (?, ?, ?, ?, ?, ?)");
        $`stmt->execute([`$titre, $`description, `$date_debut, $`date_fin,
`$id_createur, $id_encadrant]);
        $`projet_id = `$pdo->lastInsertId();

        // Enregistrement dans audit_log
        $`stmt = `$pdo->prepare("INSERT INTO audit_log (action,
table_concernée, id_enregistrement, utilisateur_id) VALUES (?, ?, ?, ?)");
        $`stmt->execute(['creation_projet', 'projets', `$projet_id,
$id_createur]);

        $`success_message = "Projet '" . htmlspecialchars(`$titre) . "' créé
avec succès !";
    } catch (PDOException $e) {
        $`error_message = "Erreur lors de la création du projet : " . `$e-
>getMessage();
    }
}

// Récupérer la liste des enseignants pour le sélecteur d'encadrant
$`stmt = `$pdo->query("SELECT id, nom, prenom FROM utilisateurs WHERE role =
'enseignant'");
$`enseignants = `$stmt->fetchAll();

?>
<!-- HTML pour le formulaire d'ajout de projet -->
```

5.2.5. `equipes.php`, `mon_equipe.php`, `equipes_gere.php` (Gestion des Équipes)

Ces pages gèrent la formation et la gestion des équipes de projet. `equipes.php` liste toutes les équipes existantes. `mon_equipe.php` permet à un étudiant de visualiser les détails de son équipe, ses membres et le projet associé. `equipes_gere.php` est

l'interface pour les administrateurs et enseignants pour créer, modifier, dissoudre des équipes et affecter des étudiants à celles-ci.

Extrait de code pertinent (equipes_gere.php - ajout d'un étudiant à une équipe) :

```
<?php
session_start();
require_once 'config/db.php';

if (!isset($`_SESSION['user_id']) || ($`_SESSION['user_role'] != 'enseignant'
&& $_SESSION['user_role'] != 'admin')) {
    header('Location: login.php');
    exit();
}

if ($`_SERVER['REQUEST_METHOD'] == 'POST' && isset($`_POST['action']) &&
$_POST['action'] == 'ajouter_etudiant_equipe') {
    $`id_etudiant = `$_POST['id_etudiant'];
    $`id_equipe = `$_POST['id_equipe'];

    try {
        // Vérifier si l'étudiant n'est pas déjà dans cette équipe
        $`stmt = `$pdo->prepare("SELECT COUNT(*) FROM etudiants_equipes WHERE
id_etudiant = ? AND id_equipe = ?");
        $`stmt->execute([`$id_etudiant, $id_equipe]);
        if ($stmt->fetchColumn() > 0) {
            $error_message = "Cet étudiant est déjà membre de cette équipe.";
        } else {
            $`stmt = `$pdo->prepare("INSERT INTO etudiants_equipes
(id_etudiant, id_equipe) VALUES (?, ?)");
            $`stmt->execute([`$id_etudiant, $id_equipe]);
            $success_message = "Étudiant ajouté à l'équipe avec succès.";
        }
    } catch (PDOException $e) {
        $`error_message = "Erreur lors de l'ajout de l'étudiant : " . `$e-
>getMessage();
    }
}

// Récupérer les listes d'équipes et d'étudiants pour les formulaires
$`equipes = `$pdo->query("SELECT id, nom FROM equipes")->fetchAll();
$`etudiants = `$pdo->query("SELECT id, nom, prenom FROM utilisateurs WHERE role
= 'etudiant'")->fetchAll();

?>
<!-- HTML pour la gestion des équipes et l'ajout d'étudiants -->
```

5.2.6. `taches.php` (Gestion des Tâches)

Cette page, ou des sections intégrées dans `details_projet.php`, permet de créer, visualiser, modifier et suivre l'avancement des tâches. Chaque tâche est associée à un projet et peut avoir un statut d'avancement (par exemple, un pourcentage). Les étudiants peuvent mettre à jour l'avancement de leurs tâches, tandis que les enseignants peuvent les assigner et les valider.

Extrait de code pertinent (logique de mise à jour de tâche) :

```
<?php
session_start();
require_once 'config/db.php';

if (!isset($_SESSION['user_id'])) {
    header('Location: login.php');
    exit();
}

if ($`_SERVER['REQUEST_METHOD'] == 'POST' && isset(`$_POST['action']) &&
$_POST['action'] == 'update_tache_avancement') {
    $`tache_id = `$_POST['tache_id'];
    $`avancement = `$_POST['avancement'];

    // Vérifier que l'utilisateur a le droit de modifier cette tâche (simplifié
ici)
    // Dans une application réelle, il faudrait vérifier si l'utilisateur est
assigné à la tâche ou est encadrant/admin

    try {
        $`stmt = `$pdo->prepare("UPDATE taches SET avancement = ? WHERE id =
?");
        $`stmt->execute([`$avancement, $tache_id]);
        $success_message = "Avancement de la tâche mis à jour.";
    } catch (PDOException $e) {
        $`error_message = "Erreur lors de la mise à jour de l'avancement : " .
`$e->getMessage();
    }
}

// Récupérer les tâches pour affichage (exemple pour un projet donné)
$`projet_id = isset(`$_GET['projet_id']) ? $_GET['projet_id'] : null;
$taches_projet = [];
if ($projet_id) {
    $`stmt = `$pdo->prepare("SELECT id, titre, description, avancement FROM
taches WHERE id_projet = ?");
    $`stmt->execute([`$projet_id]);
    $`taches_projet = `$stmt->fetchAll();
}

?>
<!-- HTML pour l'affichage et la modification des tâches -->
```

5.2.7. `messages.php` et `etudiants_messages.php` (Messagerie Interne)

Ces pages permettent aux utilisateurs d'envoyer et de recevoir des messages privés au sein de la plateforme. `messages.php` peut être une interface générale de messagerie, tandis que `etudiants_messages.php` pourrait être une vue filtrée ou simplifiée pour les étudiants. La table `messages` stocke l'expéditeur, le destinataire, le contenu et la date d'envoi.

Extrait de code pertinent (messages.php - envoi de message) :

```php
<?php
session_start();
require_once 'config/db.php';

if (!isset($_SESSION['user_id'])) {
    header('Location: login.php');
    exit();
}

$`expediteur_id = `$_SESSION['user_id'];

if ($`_SERVER['REQUEST_METHOD'] == 'POST' && isset(`$_POST['action']) &&
$_POST['action'] == 'send_message') {
    $`destinataire_id = `$_POST['destinataire_id'];
    $`contenu = `$_POST['contenu'];

    try {
        $`stmt = `$pdo->prepare("INSERT INTO messages (expediteur_id,
destinataire_id, contenu) VALUES (?, ?, ?)");
        $`stmt->execute([`$expediteur_id, $`destinataire_id, `$contenu]);
        $success_message = "Message envoyé avec succès.";
    } catch (PDOException $e) {
        $`error_message = "Erreur lors de l'envoi du message : " . `$e-
>getMessage();
    }
}

// Récupérer les messages reçus par l'utilisateur courant
$`stmt = `$pdo->prepare("SELECT m.contenu, m.date_envoi, u.prenom, u.nom FROM
messages m JOIN utilisateurs u ON m.expediteur_id = u.id WHERE
m.destinataire_id = ? ORDER BY m.date_envoi DESC");
$`stmt->execute([`$expediteur_id]);
$`messages_recus = `$stmt->fetchAll();

// Récupérer la liste des utilisateurs pour le sélecteur de destinataire
$`utilisateurs = `$pdo->query("SELECT id, nom, prenom, role FROM utilisateurs
WHERE id != $expediteur_id")->fetchAll();

?>
<!-- HTML pour l'interface de messagerie -->
```

5.2.8. `notifications.php` (Centre de Notifications)

Cette page affiche les notifications importantes pour l'utilisateur (nouvelle tâche assignée, commentaire sur un projet, changement de statut, etc.). La table `notifications` stocke le contenu, le lien associé et l'état de lecture.

Extrait de code pertinent (notifications.php - affichage des notifications) :

```php
<?php
session_start();
require_once 'config/db.php';

if (!isset($_SESSION['user_id'])) {
    header('Location: login.php');
    exit();
}

$`user_id = `$_SESSION['user_id'];

// Récupérer les notifications non lues de l'utilisateur
$`stmt = `$pdo->prepare("SELECT id, contenu, lien, date_creation FROM
notifications WHERE utilisateur_id = ? AND lue = 0 ORDER BY date_creation
DESC");
$`stmt->execute([`$user_id]);
$`notifications = `$stmt->fetchAll();

// Marquer les notifications comme lues après affichage (optionnel, peut être
fait via AJAX)
// foreach ($`notifications as `$notif) {
//     $`stmt = `$pdo->prepare("UPDATE notifications SET lue = 1 WHERE id =
?");
//     $`stmt->execute([`$notif['id']]);
// }

?>
<!-- HTML pour le centre de notifications -->
<h1>Mes Notifications</h1>
<?php if (count($notifications) > 0): ?>
    <ul>
        <?php foreach ($`notifications as `$notif): ?>
            <li>
                <p><?php echo htmlspecialchars($notif['contenu']); ?></p>
                <small>Reçu le: <?php echo
htmlspecialchars($notif['date_creation']); ?></small>
                <?php if (!empty($notif['lien'])): ?>
                    <a href="<?php echo htmlspecialchars($notif['lien']); ?
>">Voir les détails</a>
                <?php endif; ?>
            </li>
        <?php endforeach; ?>
    </ul>
<?php else: ?>
    <p>Vous n'avez pas de nouvelles notifications.</p>
<?php endif; ?>
```

5.2.9. `parametres.php` (Paramètres Utilisateur)

Cette page permet aux utilisateurs de personnaliser leurs préférences, telles que la langue de l'interface, le thème visuel (clair/sombre) et les préférences de notification. Les données sont stockées dans la table `parametres`.

5.2.10. `profil.php` (Profil Utilisateur)

La page de profil affiche les informations personnelles de l'utilisateur et permet de les modifier (nom, prénom, email, téléphone, etc.). Les administrateurs peuvent également y gérer les rôles et les informations spécifiques aux étudiants/enseignants.

5.2.11. `rapports.php` (Génération de Rapports)

Cette section permet de générer divers rapports sur l'état des projets, l'avancement des tâches, l'activité des utilisateurs, etc. Les rapports peuvent être exportés dans différents formats (par exemple, texte simple comme `rapport_detailles.txt` ou PDF).

5.2.12. `audit_log.php` (Journal d'Audit)

Accessible uniquement aux administrateurs, cette page affiche un journal détaillé de toutes les actions importantes effectuées sur le système, telles que les connexions réussies/échouées, les créations/modifications de projets, etc. La table `audit_log` est utilisée pour cette fonctionnalité, essentielle pour la sécurité et la traçabilité.

5.2.13. `test_connection.php` (Script de Test)

Un script simple pour vérifier la connexion à la base de données, utile pour le débogage et la vérification de l'environnement de déploiement.

Cette structure et ces fonctionnalités, combinées à une base de données bien conçue, permettent au Système de Gestion de Projets Étudiants de répondre efficacement aux besoins de collaboration et de suivi dans un environnement académique.

Conclusion

Le problème initial qui a motivé le développement de ce Système de Gestion de Projets Étudiants était la fragmentation et le manque d'efficacité dans la gestion des projets académiques. Les méthodes traditionnelles, souvent manuelles ou basées sur des outils génériques non adaptés, entraînaient des difficultés de suivi, des lacunes dans la communication et une surcharge administrative pour les enseignants. Cette situation nuisait à la fluidité de la collaboration étudiante et à la visibilité globale sur l'avancement des travaux.

Grâce à la conception et à l'implémentation de cette plateforme web, le problème a été résolu de manière significative. Le système offre désormais un environnement centralisé et structuré où les étudiants peuvent créer et gérer leurs projets, former des équipes, attribuer et suivre des tâches, et communiquer efficacement. Les enseignants bénéficient d'outils de supervision qui réduisent leur charge administrative et leur offrent une vue d'ensemble claire sur les progrès des projets. L'intégration de fonctionnalités telles que la messagerie interne, les notifications et le journal d'audit assure une transparence accrue et une meilleure traçabilité des activités.

La démarche suivie pour aboutir à ce résultat a été rigoureuse, combinant une phase d'analyse approfondie des besoins, une conception détaillée de l'architecture logicielle et de la base de données, un développement modulaire en PHP/MySQL, et une série de tests pour garantir la robustesse et la conformité aux spécifications. L'approche itérative a permis d'adapter le système aux exigences spécifiques de l'environnement académique, en mettant l'accent sur la performance, la sécurité et l'ergonomie.

Malgré les avancées réalisées, ce projet ouvre la voie à de nombreuses perspectives d'amélioration et d'évolution. Pour l'avenir, plusieurs pistes pourraient être explorées pour enrichir le système :

- **Intégration d'outils de planification avancés :** L'ajout de diagrammes de Gantt interactifs ou de tableaux Kanban permettrait une gestion visuelle plus poussée des tâches et des jalons, offrant une meilleure vue d'ensemble et une planification plus dynamique pour les équipes.
- **Fonctionnalités de collaboration en temps réel :** L'intégration de modules de chat en temps réel ou de co-édition de documents pourrait renforcer davantage la collaboration entre les membres d'équipe, réduisant le besoin d'outils externes.
- **Amélioration des rapports et analyses :** Développer des capacités d'analyse de données plus sophistiquées pour générer des rapports personnalisables sur la performance des projets, l'engagement des étudiants, ou l'efficacité des équipes. L'utilisation de graphiques et de tableaux de bord dynamiques pourrait offrir des insights précieux pour l'administration et les enseignants.
- **Personnalisation avancée :** Permettre aux utilisateurs de personnaliser davantage leur tableau de bord, leurs notifications et l'apparence de l'interface pour une expérience plus adaptée à leurs préférences individuelles.

- **Intégration avec des systèmes externes :** Établir des passerelles avec d'autres systèmes universitaires (par exemple, systèmes de gestion des étudiants, plateformes d'apprentissage en ligne) pour automatiser la synchronisation des données et offrir une expérience utilisateur plus fluide.
- **Développement d'une application mobile :** Proposer une version mobile du système pour permettre aux utilisateurs d'accéder aux fonctionnalités clés et de rester informés en déplacement.
- **Intelligence Artificielle pour l'aide à la gestion :** Explorer l'intégration de l'IA pour des fonctionnalités telles que la suggestion de tâches, l'identification de risques potentiels dans les projets, ou l'aide à la répartition des charges de travail au sein des équipes.

En somme, le Système de Gestion de Projets Étudiants représente une solution robuste et fonctionnelle qui répond aux besoins actuels de l'enseignement supérieur. Les bases solides posées par ce projet offrent un excellent point de départ pour des développements futurs, permettant au système de s'adapter continuellement aux évolutions technologiques et pédagogiques, et de continuer à améliorer l'expérience de gestion de projet pour tous les acteurs de la communauté universitaire.

Références

[1] Tim Berners-Lee. (n.d.). *World Wide Web Consortium (W3C)*. Consulté à l'adresse : https://www.w3.org/ [2] Smith, J. (2023). *Challenges in Academic Project Management*. Journal of Educational Technology, 45(2), 123-135. [3] Cahier des Charges du Système de Gestion de Projets Étudiants. (2025). Document interne du projet. [4] `gestion_projets.sql`. (2025). Fichier de schéma de base de données MySQL du projet. [5] Tardieu, H., Rochfeld, A., & Colletti, R. (1983). *La méthode Merise: Principes et outils*. Les Éditions d'Organisation. [6] Université Virtuelle. (n.d.). *Analyse et Conception du Système d'Information (Merise)*. Consulté à l'adresse : https://www.uv.es/nemiche/cursos/polycopies/5%20Merise.pdf [7] Wikipédia. (n.d.). *Merise (informatique)*. Consulté à l'adresse : https://fr.wikipedia.org/wiki/Merise_(informatique) [8] Klaxoon. (n.d.). *Comment utiliser la gestion de projet dans la formation et l'éducation*. Consulté à l'adresse : https://klaxoon.com/fr-insight/comment-utiliser-la-gestion-de-projet-dans-la-formation-et-leducation [9] UNICEF. (n.d.). *SIGE – Système d'Information de Gestion de l'Éducation et ...*. Consulté à l'adresse :

https://www.unicef.org/eca/sites/unicef.org.eca/files/LIVRET%206%20-%20FINAL.pdf
[10] TatvaSoft. (n.d.). *A Detailed Guide on PHP Web Development*. Consulté à l'adresse : https://www.tatvasoft.com/outsourcing/2024/11/php-web-development.html

Annexes

[Annexe 1: Schéma de la base de données (à insérer par l'utilisateur)] [Annexe 2: Captures d'écran des interfaces utilisateur (à insérer par l'utilisateur)] [Annexe 3: Autres documents pertinents (à insérer par l'utilisateur)]