Université Protestante au Congo

Faculté des Sciences Informatiques

Critères d'Acceptabilité des Projets

L2 LMD FASI

Décembre 2024

Contenu

1 Introduction Générale aux Projets Académiques : Une Expérience Décisive pour Votre Parcours

La réalisation d'un projet académique n'est pas une simple formalité : c'est un rite de passage qui symbolise la transition entre l'apprentissage théorique et la mise en pratique des compétences acquises. C'est l'occasion pour chaque étudiant de démontrer non seulement ses connaissances, mais aussi sa capacité à résoudre des problèmes concrets, à innover et à produire un travail de qualité, comparable à ce qui est attendu dans le monde professionnel.

1.1 Le Projet Académique : Une Étape Cruciale dans votre Formation

Un projet académique, qu'il s'agisse de développement logiciel, de création d'applications web ou mobiles, ou encore de systèmes électroniques avancés, est bien plus qu'un exercice pratique. Il constitue un outil pédagogique essentiel qui :

- Consolide vos compétences techniques en vous confrontant à des défis réels.
- Développe votre autonomie et votre esprit d'analyse, deux qualités indispensables dans le monde du travail.
- Renforce vos aptitudes à travailler en équipe, à collaborer efficacement et à gérer des projets dans des délais impartis.

C'est dans cette perspective que les projets de cette année sont particulièrement significatifs. Ils représentent une **opportunité unique** de sortir de votre zone de confort, d'explorer de nouvelles technologies et de démontrer vos capacités à mener à bien une mission complexe de bout en bout.

1.2 Une année Décisive : Le Moment pour prouver Votre Potentiel

Cette année académique marque une étape déterminante dans votre parcours. Le projet que vous allez réaliser n'est pas un simple exercice parmi d'autres, mais un élément clé de votre formation. Il vous prépare à affronter les exigences du monde professionnel, où la rigueur, la discipline et la qualité du travail sont les piliers de la réussite. Chaque détail,

chaque ligne de code, chaque composant de votre interface, chaque schéma technique aura une importance capitale. Votre investissement personnel, votre capacité à gérer votre temps, à faire preuve de créativité et à respecter les normes établies détermineront non seulement la réussite de ce projet, mais également la perception que vos encadrants et vos futurs employeurs auront de vous. Vous ne pouvez pas vous permettre la médiocrité.

C'est ici que vous devez montrer que vous êtes prêt à relever des défis, à surmonter des obstacles et à livrer un travail dont vous pourrez être fier.

1.3 Critères d'Acceptabilité : Les Fondations d'un Projet Réussi

Un projet académique ne se limite pas à une idée ou à une simple réalisation technique. Il doit répondre à des critères bien définis, qui garantissent sa qualité, sa fiabilité et sa pertinence. Ces critères sont vos repères tout au long du processus de développement.

1.3.1 Rigueur Technique

Chaque projet doit s'appuyer sur une base technique solide :

- Conception cohérente et modulaire : Que vous développiez un logiciel, une application mobile, ou un système électronique, votre architecture doit être claire, évolutive et bien structurée.
- Qualité du code : Votre code doit être propre, commenté, et conforme aux bonnes pratiques du langage utilisé.
- Intégration de fonctionnalités complètes : Le projet doit offrir toutes les fonctionnalités essentielles définies dans votre cahier des charges.

1.3.2 Maîtrise de l'Interface Utilisateur

- **Ergonomie et design** : L'interface doit être intuitive, attrayante et accessible. Une interface bien pensée facilite l'interaction utilisateur et valorise votre travail.
- **Accessibilité** : Votre projet doit être utilisable par tous, quel que soit le niveau d'expérience de l'utilisateur final.

1.3.3 Gestion des Données et Sécurité

Stockage et gestion des données : L'utilisation d'une base de données bien conçue est essentielle pour gérer efficacement les informations collectées, qu'elles proviennent d'une application ou de capteurs dans un projet technique.

1.3.4 Performance et Optimisation

- **Réactivité et fluidité** : Votre projet doit offrir une expérience utilisateur fluide, avec des temps de réponse rapides.
- **Optimisation des ressources** : Que ce soit en termes de mémoire, de consommation énergétique, ou de bande passante, votre projet doit être optimisé pour fonctionner de manière efficace.

1.3.5 Documentation et Présentation

Un projet réussi ne se limite pas à sa réalisation technique. La documentation est la clé pour assurer sa compréhension et son utilisation.

- **Documentation technique** : Décrivez chaque module, chaque composant, et chaque étape de développement.
- **Manuel utilisateur** : Fournissez des instructions claires pour que l'utilisateur final puisse exploiter votre projet sans difficulté.

1.3.6 Discipline et Rigueur : Les Clés de Votre Réussite

La réussite de votre projet dépendra en grande partie de votre capacité à respecter les principes établis dans ce guide. Chaque étape, chaque livrable doit être accompli avec discipline et rigueur. Ce sont ces qualités qui distinguent un projet acceptable d'un projet remarquable.

- **Respect des délais** : Le respect du calendrier de réalisation est impératif.
- **Communication et collaboration** : N'hésitez pas à solliciter l'aide de vos encadrants et à travailler en équipe lorsque cela est nécessaire.
- **Perfectionnement continu** : Soyez à l'écoute des critiques constructives et utilisez-les pour améliorer votre travail.

1.3.7 Conclusion : Un Tremplin vers l'Excellence

Votre projet académique est bien plus qu'une simple évaluation : c'est une opportunité unique de prouver vos compétences, d'apprendre, de créer, et de vous préparer à intégrer le monde professionnel. Voyez ce projet comme un tremplin vers l'excellence. C'est

l'occasion de dépasser vos limites, de démontrer votre potentiel et de poser les bases d'une carrière réussie. Prenez cette mission au sérieux, investissez-vous pleinement, et montrez que vous êtes prêt à relever les défis qui se présentent à vous.

2 Projet Site/Application Web

2.1 Description

Un projet de site ou application web est une initiative visant à concevoir, développer et déployer une plateforme accessible via un navigateur internet. Ce type de projet peut remplir diverses fonctions selon les objectifs définis, allant de la simple présentation d'informations à la fourniture de services complexes nécessitant l'interaction avec les utilisateurs.

2.1.1 Caractéristiques principales

- **Conception de l'interface utilisateur (UI/UX)** : Le projet commence par la création d'une interface conviviale et intuitive pour permettre aux utilisateurs de naviguer facilement, d'interagir avec les fonctionnalités, et d'accéder rapidement aux informations.
- **Développement du front-end** : La partie visible du site ou de l'application (le front-end) est codée en utilisant des langages comme HTML, CSS, et JavaScript, souvent assistés par des frameworks tels que React, Vue.js ou Angular.
- **Développement du back-end** : Le back-end gère la logique métier, les interactions avec la base de données et la sécurité. Il est souvent développé avec des langages et frameworks comme Python (Django, Flask), Node.js, Ruby on Rails, ou PHP (Laravel).
- **Base de données** : Le projet nécessite généralement une base de données pour stocker, gérer et récupérer les données utilisateur ou d'autres informations importantes. Des systèmes de gestion de base de données comme MySQL, PostgreSQL, ou MongoDB sont utilisés.
- **Fonctionnalités principales** :
  - **Gestion des utilisateurs** : création de comptes, authentification, et gestion des profils.
  - **Interaction utilisateur** : commentaires, likes, partages, envoi de formulaires, etc.
  - **Traitement des données** : affichage dynamique des informations selon les besoins des utilisateurs.
  - **Intégration d'API** : pour enrichir l'application avec des services externes comme la géolocalisation, les paiements en ligne, etc.
- **Sécurité** : La protection des données et la gestion des accès sont des éléments essentiels. Des techniques comme le chiffrement des données, les pare-feu applicatifs, et la gestion des sessions sécurisées sont souvent mises en œuvre.
- **Déploiement et maintenance** : Le projet se termine par le déploiement sur un serveur web ou dans le cloud, ainsi que par la mise en place d'un système de maintenance pour assurer le bon fonctionnement et l'évolution continue de la plateforme.

2.2 CRITÈRES

2.2.1 Intégration de la base de données

La base de données joue un rôle central dans la gestion des informations dynamiques et interactives du site web.

- **Connexion sécurisée** : Mise en œuvre d'une connexion sécurisée avec la base de données pour assurer l'intégrité et la confidentialité des données.
- **Opérations CRUD (Create, Read, Update, Delete)** : Implémentation complète des fonctionnalités CRUD pour permettre l'ajout, la consultation, la modification et la suppression des données affichées sur le tableau de bord de l'application.

2.2.2 Utilisation obligatoire d'un langage de programmation côté Back-end

Le développement back-end doit se faire avec un langage de programmation robuste et adapté aux interactions avec la base de données, à la gestion des utilisateurs et à la sécurité.

2.2.3 Utilisation des méthodes de conception MERISE

Les modèles de données et les schémas de la base de données doivent être conçus selon les méthodologies MERISE (modèle conceptuel, logique et physique) afin de garantir une organisation cohérente et évolutive des données.

2.2.4 Présence d'une authentification

Le système doit inclure une fonctionnalité d'authentification sécurisée permettant aux utilisateurs de se connecter avec un identifiant et un mot de passe, assurant la protection des accès.

2.2.5 Gestion des rôles et des autorisations

Mise en place d'un système de gestion des utilisateurs basé sur des rôles et des autorisations pour contrôler l'accès aux différentes sections et fonctionnalités de l'application (administrateurs, utilisateurs standards, etc.).

2.2.6 Fonctionnalités de tableaux de bord

Le tableau de bord doit fournir une interface interactive pour la gestion et l'analyse des données.

- **Visualisation des données** : Utilisation de bibliothèques comme Chart.js ou D3.js pour afficher les données sous forme de graphiques interactifs (barres, lignes, camemberts).
- **Filtres et tri** : Intégration de fonctionnalités permettant aux utilisateurs de filtrer et de trier les données en fonction de critères définis.
- **Rapports dynamiques** : Génération de rapports automatisés et personnalisés basés sur les données stockées dans la base de données.

2.2.7 Déploiement de la solution sur Internet

Le site web et sa base de données doivent être déployés en ligne pour permettre un accès distant et offrir une expérience utilisateur fluide. Ce déploiement doit garantir la sécurité des données et une haute disponibilité.

2.2.8 Gestion de la responsivité

L'interface utilisateur doit être entièrement responsive, assurant une compatibilité optimale avec différentes tailles d'écran (ordinateurs, tablettes, smartphones).

2.2.9 Développement Back-end sans Framework

Le développement back-end doit se faire sans utiliser de Framework complet, mais l'usage de bibliothèques spécialisées est autorisé pour simplifier certaines tâches (gestion des sessions, traitement des fichiers, etc.).

2.2.10 Hébergement du code sur un dépôt distant (GitHub)

Le code source du projet doit être hébergé sur une plateforme de gestion de versions comme GitHub, permettant un suivi rigoureux des modifications et facilitant le travail collaboratif.

2.2.11 Présence d'un gestionnaire de version (Git)

Utilisation de Git comme système de gestion de version pour suivre l'évolution du projet, collaborer efficacement avec l'équipe et assurer une traçabilité complète des modifications.

2.2.12 Aspects sécurité

- **Stockage sécurisé des mots de passe** : Ne jamais stocker les mots de passe en texte clair. Utiliser des algorithmes de hachage sécurisés comme bcrypt ou SHA-256 avec un sel.
- **Validation des entrées utilisateur** : Vérifier et nettoyer toutes les données saisies par l'utilisateur pour éviter les attaques par injection (SQL, XSS).
- **Protection contre les injections SQL** : Utiliser des requêtes préparées ou des ORM (Object-Relational Mapping) pour interagir avec la base de données.
- **Déconnexion sécurisée** : Implémenter une fonction de déconnexion qui détruit correctement la session de l'utilisateur.

3 Projets Applications Desktop

3.1 Description

Un projet d'application desktop est une initiative visant à concevoir, développer et déployer un logiciel destiné à être installé et utilisé sur un ordinateur personnel ou un poste de travail. Ces applications sont souvent conçues pour fonctionner hors ligne, mais elles peuvent également intégrer des fonctionnalités en ligne pour enrichir l'expérience utilisateur.

3.1.1 Caractéristiques principales

- **Conception de l'interface utilisateur (UI/UX)** : L'interface utilisateur d'une application desktop doit être intuitive, réactive et adaptée à une utilisation prolongée sur un écran d'ordinateur.
  - **Design ergonomique** : organisation claire des fenêtres, menus, et barres d'outils.
  - **Composants graphiques standards** : boutons, champs de saisie, listes déroulantes, et boîtes de dialogue pour une navigation fluide.
  - **Compatibilité multi-plateforme** : adaptation de l'interface pour les systèmes d'exploitation comme Windows, macOS, et Linux.
- **Développement de l'architecture logicielle** : Le développement d'une application desktop repose sur une architecture bien définie pour assurer sa robustesse, sa modularité et sa maintenabilité.
  - **Architecture en couches** : séparation de la logique métier, de l'interface utilisateur, et des accès aux données.
  - **Modèle MVC (Model-View-Controller) ou MVVM (Model-View-ViewModel)** pour structurer le code et faciliter les évolutions futures.
  - **Programmation orientée objet (POO)** pour organiser le code de manière modulaire et réutilisable.
- **Développement multi-plateforme ou natif** : Les applications desktop peuvent être développées en natif ou en utilisant des frameworks multi-plateformes pour assurer la compatibilité avec plusieurs systèmes d'exploitation.
  - **Langages natifs** :
    - C Sharp avec WPF ou WinForms pour Windows.
    - Swift ou Objective-C pour macOS.
    - C++ pour des performances élevées sur toutes les plateformes.
  - **Frameworks multi-plateformes** :
    - Electron (JavaScript, HTML, CSS) pour des applications web empaquetées en desktop.
    - Qt (C++) pour une compatibilité avec Windows, macOS, et Linux.
    - JavaFX pour des applications Java fonctionnant sur tous les systèmes.

- **Base de données** : Les applications desktop peuvent stocker leurs données localement ou les synchroniser avec une base de données distante.
  - **Bases de données locales** : SQLite, Microsoft Access, ou Realm pour le stockage hors ligne.
  - **Bases de données distantes** : MySQL, PostgreSQL, ou MongoDB pour la synchronisation et le partage de données en ligne.
- **Sécurité** : La sécurité est un aspect essentiel des applications desktop, surtout lorsqu'elles manipulent des données sensibles.
  - **Authentification et autorisation** pour contrôler l'accès aux fonctionnalités critiques.
  - **Validation des entrées utilisateur** pour éviter les erreurs et les vulnérabilités.
- **Déploiement et Maintenance** : L'application desktop, une fois développée, est déployée sous forme d'installateur ou de package.
  - **Installateurs natifs** : MSI pour Windows, DMG pour macOS, ou DEB/RPM pour Linux.

3.2 Critères

3.2.1 Utilisation de la Méthode de Conception MERISE

La méthode MERISE est utilisée pour la conception des applications desktop afin d'assurer une modélisation rigoureuse des données et des processus. Elle permet de :

- Analyser et structurer les besoins des utilisateurs.
- Concevoir les modèles conceptuels, logiques et physiques des données et des traitements.
- Assimiler l'impact d'une conception bien structurée sur la maintenabilité et l'évolution de l'application.

3.2.2 Critères Techniques pour les Applications Desktop

- **Architecture Logicielle** : Une architecture logicielle bien conçue est cruciale pour garantir la robustesse, la flexibilité et la maintenabilité de l'application.
  - **Structure du Code** : Utilisation de la programmation orientée objet (POO) pour organiser le code de manière modulaire, réutilisable et facile à maintenir. Cela permet également de réduire la duplication de code et d'améliorer la lisibilité.
  - **Architecture en Couches** : Implémentation d'une architecture qui sépare les différentes couches de l'application :
    - **Modèle (Model)** : Gère la logique métier et les interactions avec la base de données.
    - **Vue (View)** : Gère l'affichage des informations à l'utilisateur.

    * **Contrôleur (Controller)** : Gère les interactions entre l'utilisateur et la logique métier.

  Cette séparation permet de découpler la logique métier de l'interface utilisateur, rendant le code plus flexible et maintenable.

- **Interface Utilisateur (UI)** : L'interface utilisateur joue un rôle clé dans l'expérience de l'utilisateur. Elle doit être intuitive, réactive et ergonomique.
  - **Design et Ergonomie** : L'application doit offrir une interface attrayante et facile à naviguer. L'utilisation de composants graphiques standards tels que des boutons, menus, formulaires, listes déroulantes et barres d'outils est essentielle pour garantir une bonne prise en main.
  - **Réactivité** : L'interface doit être réactive, c'est-à-dire qu'elle doit répondre rapidement aux actions de l'utilisateur sans délai perceptible, pour offrir une expérience fluide et agréable.
- **Fonctionnalités** : Les fonctionnalités de l'application doivent répondre aux besoins des utilisateurs et offrir des interactions complètes avec les données.
  - **Gestion de Données** : L'application doit intégrer une base de données pour stocker, gérer et récupérer les données. La gestion des opérations CRUD (Create, Read, Update, Delete) est essentielle pour permettre aux utilisateurs de manipuler les informations de manière dynamique.
  - **Fonctionnalités Complètes** : L'application doit offrir toutes les fonctionnalités nécessaires au bon fonctionnement du système, qu'il s'agisse de calculs complexes, de gestion de fichiers, ou de génération de rapports.
  - **Panel d'Administration** : Une interface d'administration doit être mise en place pour permettre la gestion des utilisateurs, des rôles, des permissions, et le suivi des données critiques.
- **Sécurité** : La sécurité est un aspect fondamental des applications desktop, surtout lorsqu'elles manipulent des données sensibles.
  - **Validation des Entrées** : Toutes les entrées utilisateur doivent être validées pour éviter les erreurs, les injections de code malveillant, ou toute autre faille de sécurité. Cela inclut la vérification du format, de la longueur et du type des données saisies.
  - **Gestion des Erreurs** : Implémentation de mécanismes robustes pour gérer les erreurs et les exceptions afin d'éviter les plantages de l'application. Cela inclut des messages d'erreur clairs et des journaux de suivi pour identifier et corriger les problèmes.
  - **Authentification et Autorisation** : Mise en place d'un système d'authentification sécurisé (par exemple, via des identifiants et mots de passe chiffrés) et d'autorisation pour gérer les droits d'accès aux différentes fonctionnalités de l'application.

4 Critères Techniques pour Applications Mobiles

4.1 Architecture de l'Application

Une architecture bien définie est essentielle pour assurer la maintenabilité, la scalabilité et la stabilité de l'application mobile.

- **Structure du Code** : Le code doit être bien structuré en suivant les principes de la programmation orientée objet (POO), facilitant ainsi la modularité, la réutilisabilité et la maintenance du projet.
- **Frameworks et Outils** : Le développement peut s'appuyer sur des frameworks mobiles tels que :
  - Android avec Java ou Kotlin pour les applications natives.
  - iOS avec Swift ou Objective-C.
  - Frameworks multiplateformes comme Flutter (Dart), React Native (JavaScript) ou Ionic pour un développement cross-platform.
- **Architecture MVC ou MVVM** : L'implémentation de modèles d'architecture comme MVC (Model-View-Controller) ou MVVM (Model-View-ViewModel) permet de découpler la logique métier de l'interface utilisateur, améliorant ainsi la clarté et la gestion du code.

4.2 Interface Utilisateur (UI/UX)

L'interface utilisateur et l'expérience utilisateur sont des aspects cruciaux pour assurer l'adoption et la satisfaction des utilisateurs.

- **Design Adaptatif** : L'application doit être responsive et s'adapter parfaitement aux différentes tailles d'écran et résolutions des appareils mobiles, tels que smartphones, tablettes et phablettes.
- **Expérience Utilisateur (UX)** : L'interface doit être intuitive et facile à naviguer, offrant une expérience utilisateur fluide et agréable, tout en minimisant le nombre de clics nécessaires pour atteindre une fonctionnalité.
- **Authentification Utilisateur** : Intégration d'un système d'authentification sécurisé, pouvant inclure :
  - Authentification via email et mot de passe.
  - Connexion via réseaux sociaux (Google, Facebook).
  - Authentification biométrique (empreinte digitale ou reconnaissance faciale).

4.3 Fonctionnalités

Les fonctionnalités doivent exploiter pleinement les capacités des appareils mobiles pour offrir une valeur ajoutée aux utilisateurs.

- **Gestion de Données Mobiles** : Utilisation d'une base de données locale comme SQLite ou Room (pour Android) pour gérer les données de l'application de manière persistante sur l'appareil.
- **Intégration des Fonctionnalités Spécifiques aux Appareils Mobiles** : L'application peut intégrer des fonctionnalités natives des appareils mobiles, telles que :
  - Caméra : Pour capturer des photos ou des vidéos.
  - GPS : Pour la localisation en temps réel.
  - Accéléromètre et gyroscope : Pour détecter les mouvements.
  - Notifications Push : Pour envoyer des alertes et des rappels en temps réel aux utilisateurs.

4.4 Performance et Optimisation

Les performances de l'application doivent être optimisées pour offrir une expérience utilisateur fluide, même sur des appareils à faibles ressources.

- **Temps de Réponse** : L'application doit être optimisée pour des temps de réponse rapides, garantissant des interactions fluides et une navigation sans ralentissement.
- **Gestion de la Mémoire** : Une gestion efficace de la mémoire est essentielle pour éviter les fuites de mémoire, les ralentissements ou les crashs, surtout sur des appareils ayant des ressources limitées.
- **Optimisation des Ressources** : Réduction de la consommation de batterie, d'utilisation du processeur, et de consommation de données pour offrir une expérience optimale aux utilisateurs.

4.5 Sécurité

La sécurité des données et des utilisateurs est primordiale pour toute application mobile.

- **Chiffrement des Données** : Les données sensibles doivent être chiffrées avant d'être stockées localement ou transmises sur le réseau.
- **Protection des Sessions** : Mise en place de jetons de session sécurisés pour gérer les connexions des utilisateurs de manière sécurisée.
- **Validation des Entrées** : Toutes les entrées utilisateur doivent être validées pour prévenir les attaques comme l'injection SQL ou les scripts intersites (XSS).

4.6 Déploiement et Maintenance

Le déploiement et la maintenance sont essentiels pour garantir le succès continu de l'application.

- Déploiement via les stores d'applications (Google Play, Apple App Store).
- Mise à jour régulière pour corriger les bugs, améliorer la sécurité et ajouter de nouvelles fonctionnalités.

- Surveillance continue des performances et de l'utilisation pour garantir la stabilité de l'application.

5 Critères des Projets Électroniques

Les projets électroniques avancés en deuxième année doivent intégrer une gestion des données collectées par les capteurs et fournir des outils permettant de suivre l'évolution du projet via une interface logicielle développée par les étudiants eux-mêmes. La dimension base de données et logiciel de gestion joue un rôle essentiel dans l'organisation et la gestion du projet.

5.1 Performance Technique du Système

- **Temps de Réponse et Latence** : Le système doit fournir des réponses rapides aux capteurs et actionneurs. La latence dans les échanges de données doit être réduite au minimum pour garantir une réactivité optimale du système.
- **Scalabilité du Système** : Le système doit pouvoir intégrer de nouveaux capteurs ou actionneurs sans compromettre les performances, en permettant un ajout facile d'éléments.

5.2 Gestion des Données via Base de Données

- **Base de Données Locale** : Pour stocker et gérer les données collectées par les capteurs (température, humidité, etc.), une base de données locale (par exemple, SQLite) devra être utilisée. Cette base de données permet de garder une trace des valeurs collectées à différentes étapes, ainsi que des informations sur l'état des actionneurs.
- **Opérations CRUD (Create, Read, Update, Delete)** : L'application doit inclure les opérations CRUD pour permettre l'ajout, la lecture, la mise à jour et la suppression des données capturées par les capteurs. Ces opérations devront être intégrées directement dans l'interface de gestion pour faciliter l'analyse et le traitement des données.

5.3 Logiciel de Gestion du Projet Développé par les Étudiants

- **Développement Interne du Logiciel** : Le logiciel de gestion du projet doit être développé par les étudiants eux-mêmes. Cela inclut la création de l'interface de gestion du projet, l'intégration des données des capteurs, la gestion des utilisateurs et des ressources, ainsi que l'optimisation de l'application pour les performances.
- **Outils de Gestion de Tâches et Suivi** : Les étudiants devront développer un système pour organiser les tâches, suivre les délais et alerter les membres de l'équipe en cas de retard. Cela peut être implémenté avec des fonctionnalités comme les notifications automatiques, l'affichage d'un tableau de bord en temps réel avec les indicateurs de progrès du projet.
- **Interface Web ou Desktop** : Selon les choix des étudiants, l'interface pourra être web (utilisant des technologies comme HTML, CSS, JavaScript) ou desktop (via des frameworks comme Tkinter pour Python). L'interface devra être ergonomique et intuitive pour permettre à l'équipe de suivre facilement l'état du projet.

- **Interface de Gestion de Projet** : Une interface logicielle de gestion doit être développée par les étudiants eux-mêmes. Cette interface permettra de monitorer ou de contrôler les capteurs ou les autres composants du projet.

- **Tableau de Bord pour Suivi** : Un tableau de bord dans l'interface logicielle doit afficher en temps réel les données collectées par les capteurs, l'état des actionneurs et les alertes éventuelles. Cela permet à l'équipe de prendre des décisions informées sur le fonctionnement du projet et d'optimiser le système si nécessaire.

- **Gestion des Notifications et Alertes** : Des notifications doivent être générées dans l'interface logicielle en cas de problème détecté (par exemple, un capteur défaillant ou une erreur de communication), permettant ainsi une réponse rapide de l'équipe.

- **Rapports de Performance** : L'interface devra aussi être capable de générer des rapports détaillant la performance du système, les résultats des tests effectués et des analyses sur les données collectées. Cela permet de faire des évaluations continues sur le fonctionnement et l'efficacité du projet.

5.4 Documentation

- **Documentation Technique et Utilisateur** : Une documentation complète doit être fournie, expliquant le fonctionnement du système, des bases de données, des tests effectués et des fonctionnalités de l'interface. Un manuel d'utilisation devra expliquer comment interagir avec l'interface logicielle de gestion.