

Matière : Projet

15 février 2024

1.1 Contenu de la matière :

Le projet se déroule durant le second semestre de la deuxième année. Il consiste en la conception et la réalisation d'un projet en informatique qui se déroule dans les mêmes conditions que celles d'une entreprise

Il est décrit à travers un cahier des charges précis il peut porter sur des thèmes très variés. Il est proposé par un ou plusieurs enseignants qui jouent le rôle de client et il doit couvrir au moins deux disciplines. Il est encadré par un enseignant qui peut être en même temps le client.

Le groupe de projet, composé de 4 étudiants au minimum et de 6 au maximum, selon la taille du projet et sous la responsabilité d'un chef de projet nommé parmi les membres du groupe, doit se comporter comme une véritable équipe.

Outre le contenu technique, qui consistera en l'application des connaissances acquises pour la mise en œuvre du cycle de développement d'un petit logiciel, l'accent sera mis sur l'acquisition et l'application des aspects organisationnels et relationnels entre les membres du groupe, de l'encadreur et du client :

1. Analyse et découpage du travail (WBS, PBS,...),
2. Répartition des charges de travail entre les membres du groupe par le chef de projet,
3. Circulation de l'information entre les membres du groupe,
4. Mise en place d'un planning de travail,
5. Exposé hebdomadaire de l'avancement du projet,
6. Délivrance des livrables fixés dans la fiche de projet,
7. Rédaction d'un rapport final,
8. Exposé du travail réalisé.

1.2 Programme :

Voici un programme détaillé pour ce module, avec un horaire de 6 heures par semaine :

Semaines	Séances (de 3 heures)	Tâches
1 Introduction et Formation des équipes	1-2	<ul style="list-style-type: none"> ✓ Présentation du projet, des enseignants clients, et des attentes. ✓ Formation d'équipes(4 à 6 étudiants par équipe). ✓ Explication du rôle du chef de projet.
2 Cahier des charges	3-4	<ul style="list-style-type: none"> ✓ Sélection du projet par chaque équipe. ✓ Analyse du cahier des charges. ✓ Discussion des technologies à utiliser.
3-4 Planification et Répartition des tâches	5-6	<ul style="list-style-type: none"> ✓ Planification initiale du projet. ✓ Identification des grandes étapes de développement.
	7-8	<ul style="list-style-type: none"> ✓ Répartition des tâches entre les membres de l'équipe par le chef de projet. ✓ Début du travail sur le cahier des charges.
5-8 Développement	9-16	<ul style="list-style-type: none"> ✓ Cycle de développement du logiciel. ✓ Mise en uvre des connaissances techniques. ✓ Résolution des problèmes rencontrés.
9 Gestion de Projet	17-18	<ul style="list-style-type: none"> ✓ Suivi du planning de travail. ✓ Exposé hebdomadaire de l'avancement du projet. ✓ Gestion des obstacles et ajustements nécessaires.
10-11 Finalisation	19-22	<ul style="list-style-type: none"> ✓ Finalisation du développement. ✓ Rédaction du rapport final. ✓ Préparation de l'exposé du travail réalisé. ✓ Examen Final.
12 Présentation	23-24	<ul style="list-style-type: none"> ✓ Présentation des projets devant un jury (enseignants clients et autres).

Ce programme est conçu pour guider les étudiants à travers les différentes phases du projet, en mettant l'accent sur la gestion de projet, la collaboration d'équipe, et la mise en uvre des compétences techniques. Les séances peuvent être ajustées en fonction des besoins spécifiques de chaque groupe de projet.

1.3 Déroulement :

1.3.1 Objectif de projet :

Au cours de ces prochaines semaines, nous allons plonger dans le monde de la conception et de la réalisation de projets informatiques, simulant les conditions d'une véritable entreprise. Notre objectif est de mettre en pratique les connaissances acquises jusqu'à présent et de développer des compétences essentielles en gestion de projet.

1.3.2 Présentation du projet

: Les enseignants clients vous fourniront des détails sur les projets potentiels. Ces projets sont conçus pour couvrir divers domaines informatiques, offrant un large éventail d'options pour que chaque équipe trouve un projet aligné sur ses intérêts et compétences.

1.3.3 Expérience professionnelle :

L'expérience professionnelle est essentielle pour une future carrière. Ce projet offre l'opportunité de mettre en pratique les connaissances techniques et de développer des compétences organisationnelles et relationnelles importantes.

1.3.4 Développement de compétences :

Outre les connaissances techniques, ce projet vise également à développer des compétences organisationnelles et relationnelles. Les étudiants apprendront à gérer leur temps, à prioriser les tâches, à travailler sous pression et à interagir avec des membres d'équipe et des parties prenantes.

1.3.5 Attentes :

Nous attendons de vous une participation active, une collaboration étroite avec votre équipe, et un engagement dans toutes les phases du projet. La réussite dépendra non seulement des compétences techniques, mais aussi de la capacité à gérer efficacement le processus.

1.3.6 Suivi régulier de l'avancement :

Pour assurer la réussite du projet, un suivi régulier de l'avancement sera effectué. Les étudiants auront l'occasion de présenter leurs progrès, d'obtenir des commentaires et de faire des ajustements si nécessaire.

1.3.7 Formation d'équipes :

Vous formerez des équipes de 4 à 6 membres. La diversité des compétences au sein de chaque équipe favorisera une collaboration fructueuse. Discutez des compétences individuelles et des préférences pour choisir un projet qui convient à tous.

1.3.8 Explication du rôle du chef de projet :

Le chef de projet jouera un rôle central dans la réussite du projet. Il sera responsable de la coordination des tâches, de la communication interne, et du suivi du planning. Le chef de projet sera choisi parmi les membres de chaque équipe.

1.3.9 Travail d'équipe (teamwork) :

Le travail d'équipe est essentielle pour réaliser avec succès un petit logiciel. Le travail d'équipe permet une collaboration efficace entre les membres de l'équipe de développement, ce qui favorise une meilleure communication, une répartition des tâches claire et une utilisation efficace des compétences individuelles. Voici quelques éléments clés du travail d'équipe dans le contexte du développement de logiciels :

1. **Communication** : Une communication claire et ouverte est essentielle pour le travail d'équipe. Les membres de l'équipe doivent être en mesure de partager des informations, de discuter des idées, de poser des questions et de résoudre les problèmes ensemble. Utilisez des outils de communication appropriés, tels que des réunions régulières, des canaux de discussion en ligne, des courriels ou des outils de collaboration en temps réel pour faciliter la communication.
2. **Répartition des tâches** : Une répartition claire des tâches est nécessaire pour s'assurer que chaque membre de l'équipe sait ce qu'il doit faire. Identifiez les compétences et les forces de chaque

membre de l'équipe et attribuez les tâches en conséquence. Assurez-vous que les tâches sont équilibrées et que personne n'est surchargé ou laissé de côté. Utilisez des outils de gestion de projet pour suivre les tâches assignées et leur progression.

3. **Collaboration** : Encouragez la collaboration entre les membres de l'équipe. Favorisez le partage d'idées, la résolution de problèmes collectifs et l'apport de contributions mutuelles. Encouragez également le partage de connaissances et d'expériences pour permettre à chacun de s'enrichir des compétences des autres membres de l'équipe. L'utilisation d'outils de collaboration en ligne, tels que des dépôts de code partagés ou des wikis d'équipe (des outils de collaboration en ligne qui permettent aux membres d'une équipe de créer, d'éditer et de partager des documents et des informations de manière collaborative.), peut faciliter cette collaboration.
4. **Gestion des conflits** : Les conflits peuvent survenir dans toute équipe de développement. Il est important d'aborder les conflits de manière constructive et de chercher des solutions mutuellement bénéfiques. Encouragez l'écoute active, la compréhension des différentes perspectives et la recherche de compromis. Si nécessaire, impliquez un médiateur pour faciliter la résolution des conflits.
5. **Suivi et coordination** : Assurez-vous que les membres de l'équipe sont synchronisés et ont une vision claire de l'avancement global du projet. Utilisez des outils de gestion de projet pour suivre les progrès, identifier les problèmes potentiels et ajuster les plans si nécessaire. Organisez régulièrement des réunions d'équipe pour partager les mises à jour, discuter des obstacles et coordonner les efforts.
6. **Feedback et amélioration continue** : Encouragez les membres de l'équipe à donner leur feedback sur les processus de travail, les décisions prises et les performances individuelles. Utilisez ces retours d'information pour identifier les domaines d'amélioration et mettre en place des mesures correctives. Favorisez une culture d'amélioration continue, où les membres de l'équipe sont encouragés à apprendre de leurs erreurs et à chercher constamment à s'améliorer.

En mettant en pratique ces principes de travail d'équipe, vous pouvez créer un environnement collaboratif et productif pour réaliser votre petit logiciel. Chaque membre de l'équipe joue un rôle important dans la réussite du projet, et la synergie entre les membres contribue à la qualité globale du logiciel développé.

Conclusion : Ces deux premières séances ont jeté les bases du projet en présentant les objectifs, en détaillant les attentes, et en formant des équipes dynamiques. Le rôle crucial du chef de projet a également été clarifié. Maintenant que les équipes sont formées, vous êtes prêts à explorer les projets potentiels et à choisir celui qui vous inspire le plus. Que cette aventure commence !

Note : La proposition de projets pour chaque groupe pourrait être intégrée dans les séances suivantes, après que les équipes aient été formées. Cela permettrait une sélection plus informée en fonction des compétences et intérêts des membres de chaque équipe.

1.4 Évaluation :

1.4.1 Évaluation continue (chaque semaine)

Critère	Description
Présence	La présence est obligatoire.
Travail	Le travail doit être réalisé par groupe de 4 à 6 étudiants (sauf exceptions)
Évaluation	Chaque semaine une note est attribuée à chaque étudiant par rapport au travail réalisé durant la semaine.

1.4.2 Évaluation de l'application (à la fin de semestre)

Critère	Description
Pourcentage	Etas d'avancement par rapport au travail demandé.
Base de données	Réalisation, connexion, forme normale,
Fonctionnement	Par rapport à ce qui est demandé.
IHM	Le respect des contraintes d'IHM.
Design	La qualité des interfaces.

1.4.3 Évaluation de rapport (à la fin de semestre)

Partie	Contenu	Nombre de pages
Introduction générale	<ul style="list-style-type: none"> ✓ Contexte générale; ✓ Contexte spécifique; ✓ Définition de sujet; ✓ Organisation de travail. 	Entre 1 et 2 pages
Chapitre 01 : État de l'art	<ul style="list-style-type: none"> ✓ Etude bibliographique sur le sujet. 	Entre 5 et 7 pages
Chapitre 02 : Cahier de charge	<ul style="list-style-type: none"> ✓ Cahier de charge; ✓ Planification : <ul style="list-style-type: none"> ✚ Identification de tâches; ✚ Le réseau PRET; ✚ Diagramme de GANTT. 	Entre 4 et 8 pages
Chapitre 03 : Conception	<ul style="list-style-type: none"> ✓ Diagramme de cas d'utilisation; ✓ Diagrammes des séquences; ✓ Diagramme de classes. 	Entre 5 et 8 pages
Chapitre 04 : Réalisation	<ul style="list-style-type: none"> ✓ Définition des outils : <ul style="list-style-type: none"> ✚ Matériels; ✚ Logiciels : <ul style="list-style-type: none"> ◆ Environnement de développement; ◆ Langages; ◆ SGBD; ◆ Framework; ✓ Interfaces et descriptions. 	Entre 7 et 10 pages
Conclusion générale	<ul style="list-style-type: none"> ✓ Travail réalisé; ✓ Avantages/limites; ✓ Difficultés rencontrés lors de développement; ✓ Perspectives. 	Entre 1 et 2 pages
Référence	<ul style="list-style-type: none"> ✓ Bibliographie. 	1 page

1.5 Cahier des charges et Planification :

L'analyse du cahier des charges est une étape cruciale pour comprendre pleinement les attentes du projet. Voici comment vous pouvez approfondir cette analyse :

1. Décomposition en Sections Clés :
 - ✚ Comprenez le contexte global du projet.

- ✦ Identifiez les parties prenantes et leurs rôles
- 2. Objectifs du Projet :
 - ✦ Définissez clairement les résultats attendus.
 - ✦ Assurez-vous que les objectifs sont mesurables et alignés sur les besoins de l'entreprise.
- 3. Exigences Fonctionnelles :
 - ✦ Listez toutes les fonctionnalités attendues du logiciel.
 - ✦ Détaillez les actions spécifiques que le logiciel doit accomplir.
 - ✦ Utilisez des verbes d'action pour décrire les fonctionnalités. Organisez-les en catégories logiques (par exemple, fonctionnalités de base, fonctionnalités avancées)
- 4. Exigences Non Fonctionnelles :
 - ✦ Identifiez les contraintes de performance, de sécurité, de convivialité, etc.
 - ✦ Considérez les aspects de qualité, tels que la maintenabilité, la scalabilité, etc.
 - ✦ Spécifiez les conditions auxquelles le système doit répondre.
- 5. Contraintes Techniques :
 - ✦ Mentionnez les technologies spécifiques à utiliser.
 - ✦ Spécifiez les versions minimales requises des logiciels et des langages de programmation.
- 6. Planification :
 - ✦ Identifiez les grandes étapes du projet.
 - ✦ Établissez une chronologie réaliste pour chaque phase.
- 7. Livraisons Attendues :
 - ✦ Précisez les livrables attendus à chaque étape.
 - ✦ Clarifiez les attentes en termes de documentation.

1.5.1 Décomposition en Sections Clés :

- ✓ **Compréhension du Contexte Global du Projet :**
 - ✦ Explorez le domaine dans lequel le logiciel sera utilisé.
 - ✦ Identifiez les tendances actuelles et les défis du secteur concerné.
- ✓ **Identification des Parties Prenantes et Leurs Rôles :**
 - ✦ Recensez toutes les parties prenantes, internes et externes.
 - ✦ Définissez clairement les rôles et responsabilités de chacune.

Exemple : Si le projet concerne un système de gestion des ressources humaines (GRH), les parties prenantes pourraient inclure les employés, les responsables RH, et les administrateurs système. Les rôles pourraient être définis comme suit :

- ✦ **Employés :** Utilisation du système pour demander des congés et accéder à leurs informations personnelles.
- ✦ **Responsables RH :** Validation des demandes de congés, gestion des dossiers du personnel.
- ✦ **Administrateurs système :** Maintenance et sécurité du système.

1.5.2 Objectifs du Projet :

✓ **Définition Claire des Résultats Attendus :**

- + Énumérez les livrables tangibles attendus à la fin du projet.
- + Précisez les résultats attendus en termes de fonctionnalités et de performances.

Exemple : Dans le cas d'un projet de développement d'une application mobile de suivi des dépenses, les résultats attendus pourraient inclure une application fonctionnelle, une interface utilisateur intuitive, et des fonctionnalités de suivi des dépenses avec des rapports détaillés.

✓ **Objectifs Mesurables Alignés sur les Besoins de l'Entreprise :**

- + Énoncez des objectifs spécifiques, mesurables, atteignables, pertinents et temporels (objectifs SMART).
- + Assurez-vous que ces objectifs répondent aux besoins globaux de l'entreprise.

Exemple : Si l'entreprise vise à améliorer l'efficacité opérationnelle, un objectif mesurable pourrait être de réduire le temps moyen de traitement des demandes internes de 20% d'ici la fin de l'année.

1.5.3 Exigences Fonctionnelles :

✓ **Liste de Toutes les Fonctionnalités Attendues :**

- + Énumérez de manière exhaustive toutes les actions que le logiciel doit être capable d'accomplir.
- + Organisez ces fonctionnalités en catégories logiques.

Exemple : Pour un système de gestion de projet, les fonctionnalités de base pourraient inclure la création de tâches, l'affectation de ressources, et la gestion des délais. Les fonctionnalités avancées pourraient comprendre des outils de reporting avancés, des notifications en temps réel et une intégration avec d'autres outils de collaboration.

Note : Les exemples sont génériques et peuvent être adaptés en fonction de la nature spécifique de votre projet.

Détails des Actions Spécifiques :

1. Fonctionnalité : Gestion des Utilisateurs

- + Description : Permettre aux administrateurs d'ajouter, modifier et supprimer des utilisateurs.
- + Verbes d'Action : Ajouter, modifier, supprimer.

2. Fonctionnalité : Création de Tâches

- + Description : Les utilisateurs doivent pouvoir créer des tâches assignées à des membres spécifiques de l'équipe.
- + Verbes d'Action : Créer, assigner.

3. Fonctionnalité : Suivi du Temps

- + Description : Permettre aux utilisateurs de saisir le temps passé sur chaque tâche.
- + Verbes d'Action : Saisir, enregistrer.

4. Fonctionnalité : Génération de Rapports

- + Description : Possibilité de générer des rapports mensuels sur les performances de l'équipe.
- + Verbes d'Action : Générer, visualiser.

Exemple : Pour la fonctionnalité de "Génération de Rapports", un administrateur pourrait générer un rapport mensuel qui récapitule les heures travaillées par chaque membre de l'équipe, les tâches accomplies, et les retards éventuels.

1.5.4 Exigences Non Fonctionnelles :

✓ Identification des Contraintes :

✚ Contraintes de Performance :

- ✚ Précisez les exigences de temps de réponse, de capacité de traitement, etc.

Exemple : Le système doit permettre à 1000 utilisateurs simultanés sans diminution significative des performances.

✚ Contraintes de Sécurité :

- ✚ Définissez les exigences de sécurité des données et d'accès au système.

Exemple : Toutes les données sensibles doivent être cryptées lorsqu'elles sont stockées et transitent.

✓ Convivialité (Usabilité) :

- ✚ Énoncez les attentes en termes d'expérience utilisateur et d'interface utilisateur.

Exemple : L'interface utilisateur doit être intuitive et nécessiter une formation minimale pour les nouveaux utilisateurs.

✓ Qualité :

- ✚ Considérez des aspects tels que la maintenabilité, la fiabilité et la disponibilité.

Exemple : Le logiciel doit être conçu de manière modulaire pour faciliter les mises à jour futures.

✓ Scalabilité :

- ✚ Spécifiez la capacité du système à évoluer avec l'ajout d'utilisateurs ou de fonctionnalités.

Exemple : Le système doit pouvoir gérer une augmentation de 50% du nombre d'utilisateurs au cours de la première année.

On peut organiser les exigences non fonctionnelles comme suite :

✓ Critères de Performance :

1. Temps de Réponse :

- ✚ Critère : Le système doit répondre aux requêtes des utilisateurs en moins de 2 secondes.

2. Capacité Concurrentielle :

- ✚ Critère : Le logiciel doit prendre en charge simultanément au moins 100 utilisateurs actifs.

✓ Conditions de Sécurité :

1. Cryptage des Données :

- ✚ Condition : Toutes les données sensibles stockées dans la base de données doivent être cryptées.

2. Contrôle d'Accès :

- ✚ Condition : Seuls les utilisateurs autorisés peuvent accéder aux informations confidentielles du projet.

Exemple : Pour le critère de performance "Temps de Réponse", le système doit garantir que, lorsqu'un utilisateur effectue une action (comme la génération d'un rapport), la réponse du système est obtenue en moins de 2 secondes pour assurer une expérience utilisateur fluide.

Ces exemples illustrent comment les exigences fonctionnelles se concentrent sur les actions spécifiques du logiciel, tandis que les exigences non fonctionnelles définissent les critères et conditions auxquels le système doit répondre en termes de performances, de sécurité, etc.

1.5.5 Contraintes Techniques :

- ✓ Technologies Spécifiques à Utiliser :
 - ✦ Mentionnez les langages de programmation, les bases de données, etc.
Exemple : Le développement sera effectué en utilisant le langage Python avec une base de données MySQL.
- ✓ Versions Minimales Requises :
 - ✦ Spécifiez les versions minimales des logiciels nécessaires pour le développement et l'exécution.
Exemple : La version minimale requise du navigateur pour accéder à l'application est Google Chrome version 70.

1.5.6 Planification :

- ✓ Identification des Grandes Étapes :
 - ✦ Enumérez les principales phases du projet, de la conception à la livraison.
Exemple : 1) Analyse des besoins, 2) Conception du système, 3) Développement, 4) Tests, 5) Déploiement.
- ✓ Chronologie Réaliste pour Chaque Phase :
 - ✦ Établissez des échéanciers pour chaque phase du projet.
Exemple : La phase de développement devrait durer six semaines, avec des itérations de tests planifiées toutes les deux semaines.

1.5.7 Livraisons Attendues :

- ✓ Livrables à Chaque Étape :
 - ✦ Précisez les éléments concrets qui seront produits à chaque étape du projet.
Exemple : À la fin de la phase de conception, un document de conception détaillé sera livré.
 - ✓ Attentes en Termes de Documentation :
 - ✦ Clarifiez les attentes concernant la documentation du code, les manuels d'utilisation, etc.
Exemple : La documentation complète du code source devra être soumise avant la phase de test final.
- Note** : Ces exemples sont génériques et peuvent être ajustés en fonction des spécificités de votre projet.

Vision du Projet :

- ✓ Compréhension de la Vision Globale du Projet :
 - ✦ Vision : Faciliter la collaboration au sein de l'équipe de développement en fournissant une plateforme centralisée pour la gestion de projets.
 - ✦ Objectif Global : Le logiciel vise à améliorer l'efficacité opérationnelle en optimisant la coordination des tâches, la communication et le suivi du temps.
- ✓ Contribution aux Objectifs de l'Entreprise :
 - ✦ Alignement Stratégique : En soutenant une gestion de projet plus efficace, le logiciel contribue à la réalisation des objectifs stratégiques de l'entreprise, tels que l'optimisation des processus et l'amélioration de la productivité.
 - ✦ Impact sur la Performance Globale : En automatisant les processus de gestion de projet, l'entreprise s'attend à une utilisation plus efficace des ressources, une réduction des retards et une amélioration globale de la performance.

Bénéfices Attendus :

- ✓ Avantages pour les Utilisateurs et l'Entreprise :
 - ✦ Utilisateurs : Les membres de l'équipe bénéficieront d'une meilleure visibilité sur les tâches, d'une communication simplifiée et d'un suivi précis du temps, améliorant ainsi leur productivité.
 - ✦ Entreprise : L'entreprise anticipa une réduction des coûts opérationnels, une gestion plus efficace des projets, et une augmentation de la satisfaction client grâce à des livraisons plus rapides.
- ✓ Indicateurs Clés de Succès :
 - ✦ Taux d'Adoption : Mesurer le pourcentage d'utilisateurs actifs dans le système après le déploiement initial.
 - ✦ Réduction des Retards : Suivre la diminution du nombre de projets en retard après la mise en place du logiciel.
 - ✦ Satisfaction Utilisateur : Recueillir des retours réguliers des utilisateurs pour évaluer la satisfaction globale avec le logiciel.

Mesures de Performance :

- ✓ Critères d'Évaluation du Succès :
 - ✦ Performance du Système : Un temps de réponse moyen inférieur à 2 secondes pour toutes les actions effectuées dans le logiciel.
 - ✦ Fiabilité : Le système doit être opérationnel 99,9% du temps, minimisant ainsi les interruptions de service.
 - ✦ Taux d'Utilisation : Un taux d'utilisation constant de 80% ou plus des fonctionnalités du logiciel.
- ✓ Métriques Claires :
 - ✦ Nombre de Projets en Retard : Avant et après le déploiement du logiciel, mesurer la diminution des projets en retard.
 - ✦ Feedback Utilisateur : Recueillir des évaluations régulières des utilisateurs sur l'expérience utilisateur, la facilité d'utilisation et la satisfaction globale.
 - ✦ Réduction des Coûts Opérationnels : Comparer les coûts opérationnels liés à la gestion de projet avant et après la mise en uvre du logiciel.

Exemple : Si, avant le projet, 30% des projets étaient en retard, la mesure de performance serait de réduire ce taux à 15% ou moins après la mise en uvre du logiciel de gestion de projet.

1.6 Le cycle de vie d'un logiciel

Le cycle de vie d'un logiciel représente l'ensemble du processus de développement, de déploiement et de maintenance d'un logiciel, de sa conception initiale à sa fin de vie. Il est généralement présenté comme une série d'étapes guidant les différentes activités nécessaires pour produire un logiciel fonctionnel. Ce cycle comprend des activités de définition, de production, de livraison et de maintenance, exécutées séquentiellement pour concevoir, développer, déployer et maintenir un logiciel répondant aux besoins des utilisateurs tout au long de son cycle de vie. Chaque activité contribue à assurer la qualité et la fiabilité du logiciel. Les différentes phases associées au cycle de vie d'un logiciel :

1.6.1 Phase de définition :

1. **Planification** : Cette activité implique l'identification des objectifs du projet, l'allocation des ressources, l'estimation des coûts et des délais, et la création d'un plan de projet détaillé.
⇒ Le document résultat de l'activité de planification est le rapport de planification.
2. **Spécification** : Au cours de cette activité, les exigences fonctionnelles et non fonctionnelles (contraintes) du logiciel sont recueillies, analysées et documentées de manière à définir clairement ce que le logiciel doit faire et comment il doit le faire.
⇒ Les documents associés à cette activité sont le cahier des charges, les modèles d'analyse, le plan de tests et le prototype.

1.6.2 Phase de développement :

1. **Conception** : Cette activité consiste à concevoir l'architecture logicielle et les différents composants du système, en déterminant comment ils interagiront les uns avec les autres pour répondre aux exigences spécifiées.
⇒ Les documents produits de l'activité de conception sont les modèles de conception, le prototype, le plan de tests global et le plan de tests par module.
2. **Implémentation** : Pendant cette activité, le code source du logiciel est écrit et les composants logiciels sont développés en suivant les spécifications et les conceptions définies précédemment.
⇒ Les documents produits de cette activité sont les dossiers de programmation, le code source commenté et le prototype.
3. **Vérification** : Il s'agit de l'activité de test, où le logiciel est testé pour s'assurer qu'il fonctionne correctement et qu'il répond aux exigences spécifiées. Cela inclut des tests unitaires, des tests d'intégration, des tests système, des tests de régression, etc.
⇒ Le document associé à cette activité est le rapport de tests.

1.6.3 Phase de Livraison :

1. **Installation et déploiement** : Après que le logiciel a été développé et testé avec succès, il est installé et déployé dans l'environnement de production où il sera utilisé par les utilisateurs finaux.
2. **Conversion des données et formation** : Si nécessaire, les données existantes peuvent être converties pour être compatibles avec le nouveau système. De plus, une formation peut être dispensée aux utilisateurs finaux pour leur apprendre à utiliser efficacement le logiciel.

1.6.4 Phase de maintenance :

1. **Maintenance corrective** : Cette activité consiste à corriger les défauts ou les bugs découverts dans le logiciel après son déploiement.
2. **Maintenance évolutive** : Il s'agit d'ajouter de nouvelles fonctionnalités ou d'améliorer les fonctionnalités existantes du logiciel pour répondre aux besoins changeants des utilisateurs ou du marché.
3. **Maintenance préventive** : Cette activité consiste à prendre des mesures pour éviter les problèmes potentiels à l'avenir, tels que la révision et l'optimisation du code, la mise à jour des composants logiciels, etc.

1.7 Les thèmes proposés :

Thèmes 01	
Intitulé	Réalisation d'une application de suivi des projets des ingénieurs.
Domaine	Programmation JAVA, Système d'information, Bases de données.
Mots clés	SQL, UML, JAVA.
Objectifs et démarche	L'objectif de ce projet est la conception et le développement d'une application de gestion des projets de deuxième année ingénieur en utilisant UML pour la conception, Java pour le développement des interfaces et MySQL pour la gestion de base de données.
Outils et méthodes	MySQL, StartUML, JAVA, NetBeans ou Eclipse.
Fonctionnalités attendues	<p>L'application doit avoir :</p> <ol style="list-style-type: none"> Gestion des groupes : ajout, suppression et modification. Gestion des sujets : ajout, suppression et modification. <ul style="list-style-type: none"> ✓ Un sujet est connu par son titre, sa description, son domaine, ses mots clés, ses outils de développement, ... Gestion des étudiants : ajout, suppression et modification. <ul style="list-style-type: none"> ✓ Un étudiant est connu par un numéro d'inscription, le nom, le prénom, la date de naissance. ✓ Un étudiant est appartient à un seule groupe et une seule équipe. Gestion des séances : ajout, suppression et modification. <ul style="list-style-type: none"> ✓ Une séance est connue par sa date, son horaire, la salle et le groupe. Suivi des équipes : <ul style="list-style-type: none"> ✓ Création des équipes : une équipe est appartient à un seul groupe et contient au maximum 6 étudiants. ✓ Affectation des responsabilités : une équipe doit contenir un chef qui représente le chef de projet. ✓ Affectation des sujets : une équipe ne peut avoir qu'un seul sujet. Suivi des étudiants : <ul style="list-style-type: none"> ✓ Évaluation continue de chaque équipe par séance à savoir : <ul style="list-style-type: none"> + L'état d'avancement de rapport + L'état d'avancement de l'application + Une note pour chaque séance. + Une remarque pour chaque séance. ✓ Évaluation de rapport final et de l'application finale de chaque équipe. Établir des statistiques semestrielles pour chaque étudiant, chaque sujet et chaque groupe.

Thèmes 02	
Intitulé	Réalisation d'une application de gestion de stock.
Domaine	Programmation JAVA, Système d'information, Bases de données.
Mots clés	SQL, UML, JAVA.
Objectifs et démarche	L'objectif ciblé dans ce projet est la conception et le développement d'une application de gestion de stock en utilisant UML pour la conception, Java pour le développement des interfaces et MySQL pour la gestion de base de données.
Outils et méthodes	MySQL, StartUML, JAVA, NetBeans ou Eclipse.
Fonctionnalités attendues	<p>L'application doit avoir :</p> <ol style="list-style-type: none"> Gestion des produits et des familles de produits : ajout, archivage et modification. Suivi des clients et fournisseurs : ajout, suppression et modification. Gestion des entrées/sorties : <ul style="list-style-type: none"> ✓ Entrées vers stock : modification des quantités. ✓ Sorties de stock : modification des quantités. ✓ Gérer les alarmes : afficher une alarme pour un produit lorsque sa quantité atteint un certain seuil. Etablir des statistiques quotidiennes et annuelles.

Thèmes 03

Intitulé	Un site web de vente par internet.
Domaine	Développement Web, Bases de données.
Mots clés	Développement Web, SQL, UML, JavaScript, Laravel.
Objectifs et démarche	L'objectif de ce projet est la conception et le développement d'un site de vente par internet. Ce site sera un intermédiaire entre le vendeur et le client. Cette application permet de maîtriser les différentes étapes du développement d'applications web.
Outils et méthodes	Laravel, Bootstrap, Editeur code PHP, Outil d'administration de bases de données.
Fonctionnalités attendues	<p>L'application doit avoir :</p> <ol style="list-style-type: none"> Espace receveur de site : <ul style="list-style-type: none"> ✓ Inscription d'un receveur. ✓ Connexion d'un receveur inscrit après authentification. ✓ Le receveur peut parcourir tous les articles de site afin de choisir l'article qu'il veut vendre. ✓ Le site génère un lien pour accéder au formulaire de commande de ce article (ce lien est spécifique au article choisit et au receveur en même temps). ✓ Le receveur doit faire des publicités dans d'autres espaces de commerce ou réseaux sociaux (exemple : facebook, instagram, ouedkniss,) en mettant ce lien pour la commande de l'article. ✓ Le receveur doit recevoir des points à chaque vente qui arrive de ses publicités (bien sûr de lien qui a été généré pour lui par le site). ✓ A tout instant le receveur peut demander la conversion de leurs points en argents. Espace administrateur du site : <ul style="list-style-type: none"> ✓ Connexion après authentification; ✓ Suivi des receveurs (activation des inscriptions, suppression) ✓ Mise à jour des catégories d'articles (ajout, suppression) ✓ Mise à jour des articles (ajout, suppression). ✓ La gestion des demandes de conversion des points des receveurs en argents et les envoyer. Gestion de panier d'un client : <ul style="list-style-type: none"> ✓ Mise à jour du panier (ajout d'articles, modification, suppression et annulation du panier) ✓ Établissement de la commande; ✓ Validation/annulation de la commande.

Thèmes 04

Intitulé	Développement d'une application mobile pour la création et la manipulation de qcm.
Domaine	Application android.
Mots clés	UML, application android, Quiz.
Objectifs et démarche	L'objectif ciblé dans ce projet est la conception et le développement d'une application android pour saisir et utiliser un qcm. L'utilisateur (server) final doit pouvoir créer son qcm et le sauvegarder pour qu'il soit par la suite utilisé par d'autres utilisateurs (clients).
Outils et méthodes	FireBase, StartUML, JAVA, Android Studio, Flutter, vsCode.
Fonctionnalités attendues	<p>L'application doit prendre en compte deux types d'utilisateur :</p> <ol style="list-style-type: none"> Un utilisateur (server) qui crée le qcm, dans ce cas il doit pouvoir : <ul style="list-style-type: none"> ✓ Saisir les questions et leurs réponses avec le choix de la bonne réponse pour chaque question. ✓ Mettre à jour son qcm. ✓ Lancer la session de qcm. ✓ En fin de session, il envoie les notes aux clients après avoir récupérer leurs réponses. Un utilisateur (client) qui utilise le qcm, dans ce cas il doit pouvoir : <ul style="list-style-type: none"> ✓ Répondre au qcm en choisissant une réponse parmi celles proposées. ✓ En fin de session de partage, il récupère la note d'évaluation de leurs réponses. Etablir des statistiques quotidiennes et annuelles.

Thèmes 05

Intitulé	Outil d'aide à l'adaptation posologique de médicaments.
Domaine	Programmation mobile.
Mots clés	Programmation mobile, Android, Bases de données.
Objectifs et démarche	L'objectif de ce projet est la conception et le développement d'une application mobile qui d'aide à l'adaptation posologique de médicaments utilisant UML pour la conception, Java et Fxml pour le développement des interfaces et SQLite pour la gestion de base de données.
Outils et méthodes	SQLite, StartUML, JAVA, Android Studio, Flutter, vsCode.
Fonctionnalités attendues	<p>Pour administrer certains types de médicaments, le médecin demande aux patients de faire des bilans médicaux (analyses au laboratoire) parmi les trois paramètres : la clairance rénale, la bilirubine, la tgo/tgp (selon le médicament). Selon les valeurs de ces paramètres, le médecin décidera de :</p> <ul style="list-style-type: none"> ✓ Administre la dose complète du médicament, si les bilans sont normaux ✓ Ne pas administrer le médicament parce qu'il est contre-indiqué dans le cas du patient ✓ Adapter la dose selon les valeurs des bilans. <p>Exemples de médicaments : (les valeurs suivantes ne sont pas toutes correctes mais sont prises pour exemple)</p> <p>1. Médicament : cisplatine</p> <ul style="list-style-type: none"> ✓ Bilan demandé : clairance rénale ✓ Dose à administrer : <ul style="list-style-type: none"> + si la clairance rénale ≤ 30 ml/min \Rightarrow la cisplatine est contre-indiquée + si la clairance rénale ≥ 60 ml/min \Rightarrow on administre la dose complète (la valeur de la dose complète est connue par les médecins, on n'a pas à la préciser) + si 30 ml/min $<$ la clairance rénale < 60 ml/min \Rightarrow on administre 50% de la dose complète <p>2. Médicament : acide zolidronique</p> <ul style="list-style-type: none"> ✓ Bilan demandé : clairance rénale ✓ Dose à administrer : <ul style="list-style-type: none"> + si la clairance rénale ≤ 30 ml/min \Rightarrow l'acide zolidronique est contre-indiqué + si la clairance rénale ≥ 60 ml/min \Rightarrow on administre la dose complète + si 30 ml/min $<$ la clairance rénale < 40 ml/min \Rightarrow on administre 3 mg d'acide zolidronique + si 40 ml/min $<$ la clairance rénale < 50 ml/min \Rightarrow on administre 3.3 mg d'acide zolidronique + si 50 ml/min $<$ la clairance rénale < 60 ml/min \Rightarrow on administre 3.5 mg d'acide zolidronique <p>3. Médicament : capecitabine</p> <ul style="list-style-type: none"> ✓ Bilan demandé : clairance rénale, bilirubine, tgo/tgp ✓ Dose à administrer : <ul style="list-style-type: none"> + si la bilirubine ≥ 60 ml/min \Rightarrow la capecitabine est contre-indiquée + si la bilirubine < 60 ml/min \Rightarrow on administre la dose complète + si tgo/tgp ≥ 55 ml/min \Rightarrow la capecitabine est contre-indiquée + si tgo/tgp < 55 ml/min \Rightarrow on administre la dose complète + si 30 ml/min $<$ la clairance rénale < 50 ml/min \Rightarrow on administre 25% de la dose complète + si la clairance rénale ≥ 60 ml/min \Rightarrow on administre la dose complète + si la clairance rénale ≤ 30 ml/min \Rightarrow la capecitabine est contre-indiquée. <p>L'application doit avoir :</p> <ul style="list-style-type: none"> ✓ Gestion de la BD de médicaments (ajout, recherche) ✓ Sélection d'un médicament dans la base des médicaments, puis déduire la dose à administrer, suivant le bilan du malade.

Thèmes 06

Intitulé	Développement d'une plateforme collaborative pour artistes.
Domaine	Développement Web, Bases de données.
Mots clés	Développement Web, SQL, UML, JavaScript, Laravel.
Objectifs et démarche	L'objectif de ce projet est de créer une plateforme en ligne où les artistes peuvent collaborer, partager des idées, et exposer leurs uvres. L'accent sera mis sur la facilité d'utilisation et la promotion de la créativité.
Outils et méthodes	Laravel, Bootstrap, Editeur code PHP, Outil d'administration de bases de données.
Fonctionnalités attendues	<p>L'application doit avoir :</p> <ol style="list-style-type: none"> Espace artiste : <ul style="list-style-type: none"> ✓ L'artiste doit pouvoir créer un compte avec des informations de base. ✓ Le profil d'artiste peut inclure des catégories spécifiques telles que peinture, sculpture, photographie, etc. ✓ Les utilisateurs doivent pouvoir se connecter à leurs comptes en utilisant des identifiants sécurisés. ✓ L'artiste peut initier des projets collaboratifs et inviter d'autres membres à y participer. ✓ Les fonctionnalités de discussion en temps réel doivent être intégrées pour faciliter la communication. ✓ L'artiste peut partager des brouillons, des croquis, et des étapes de leur processus créatif avec d'autres utilisateurs. ✓ Chaque artiste a une galerie virtuelle pour exposer ses uvres. ✓ Les uvres peuvent être organisées par catégories et accompagnées de descriptions. ✓ Les utilisateurs peuvent évaluer et commenter les uvres exposées. ✓ Les commentaires doivent être modérés pour assurer un environnement respectueux. Espace administrateur : <ul style="list-style-type: none"> ✓ Connexion après authentification ; ✓ Suivi des utilisateurs (activation des inscriptions, suppression) ✓ Mise à jour des catégories (ajout, modification, suppression) ✓ Mise à jour des des uvres, des brouillons, des croquis, des commentaires, et des étapes (ajout, modification, suppression). ✓ Gérer la plateforme chargés de la maintenance, de la modération, et du support technique. Espace visiteur : <ul style="list-style-type: none"> ✓ Explorer et apprécier les uvres exposées sur la plateforme. ✓ Discuter en temps réel avec les artistes. ✓ Commenter les uvres exposées. respectueux.