Introduction :

L’entité SoftwareTesting, qui fait partie de C&PS/SIG au sein d’Attijariwafa bank, est une unité qui s’occupe de tout ce qui concerne la qualité des différents projets de la banque, elle s’occupe des tests, l’automatisation…

Contexte et problématique :

L’entité software testing est responsable de la qualité des applications développées au sein de la banque, cependant, il manque une solution centralisée qui permet de gérer efficacement les demandes des gestionnaires des projets de différents domaines, et publier des informations sur l’entité, ainsi que la présentation de ses réalisations.

Ajoutons à cela le manque d’informations pour les nouveaux collaborateurs sur le testing en général.

Pour demander un service de l’entité **SoftwareTesting,** ça se fait d’une maniére traditionnelle, en demandant directement de la responsable ou via un email ou autre support de communication, ce qui peut causer un retard dans l’opération.

Et pour répondre à ces besoins, nous proposons la mise en place d’un portail web intuitif et complet qui sera un point d’accès unique pour les différents collaborateurs.

Étude d’existence :

Au sein de l’entité SoftwareTesting, il n’existe aucune solution centralisée qui permet de gérer efficacement les demandes des autres entités, à part les support de communication traditionnelles , ou directement

Etude des besoins :

1. Parties prenantes principales :

**Les ressources de l’entité SoftwareTesting :** Les testeurs qui s’occupent de la réception des demandes et leur traitement, ils déposent leurs réalisations ainsi que la mise à jour de l’état de la demande.

**Ressources projets :** demandent des services de test et de qualité de l’entité SoftwareTesting, suivre l’état d’avancement de leurs demandes.

**Collaborateurs techniques :** consulter les réalisations de l’entité

1. Les besoins fonctionnelles :

* Gestion des demandes : gérer efficacement les demandes :
* Soumettre un demande via un formulaire dynamique qui comporte toutes les informations nécessaires (Sujet de la demande, type de la demande, durée estimée du service, Priorité (urgent ..), contexte et objectifs de la mission, délai, outils à utiliser, nombre de ressources à affecter .

Introduction 7486135931842611

Contexte et problématique

Étude de faisabilité

Analyse technique

Analyse économique

Analyse organisationnelle

Étude des besoins

Description des fonctionnalités clés

Besoins techniques

User stories

Planification préliminaire

Diagramme de Gantt

Architecture logicielle

Conclusion

**Gestion des réalisations**

* **Acteurs** : Collaborateur Software Testing.
* **Étapes** :
  1. L'utilisateur s'authentifie.
  2. Accède à la section des réalisations via la barre de navigation.
  3. Ajoute une nouvelle réalisation en remplissant un formulaire avec les détails nécessaires (titre, description, date, catégorie).
  4. Télécharge les fichiers associés (documents, images).
  5. Valide l'ajout et reçoit une confirmation de la publication.
* **Résultat attendu** : La réalisation est ajoutée et visible dans la bibliothèque documentaire.

**2. Recherche dans la bibliothèque documentaire**

* **Acteurs** : Collaborateur externe, Gestionnaire de projet.
* **Étapes** :
  1. L'utilisateur s'authentifie.
  2. Accède à la section "Bibliothèque documentaire".
  3. Utilise la barre de recherche pour entrer un mot-clé ou sélectionne des filtres (catégorie, date, auteur).
  4. Lance la recherche et visualise les résultats pertinents.
  5. Télécharge le document ou consulte son aperçu.
* **Résultat attendu** : Les résultats sont affichés rapidement et le document est accessible.

**3. Interaction avec le chatbot**

* **Acteurs** : Tous les utilisateurs.
* **Étapes** :
  1. L'utilisateur clique sur l'icône du chatbot dans le coin inférieur droit de l'écran.
  2. Pose une question sur un sujet précis (par exemple : "Comment soumettre une demande ?").
  3. Le chatbot analyse la requête et propose des réponses pertinentes ou suggère des articles/documentations associés.
  4. Si la question est complexe, le chatbot offre la possibilité d'escalader la demande à un expert.
* **Résultat attendu** : Réponse immédiate ou escalade vers un expert pour un suivi approfondi.

**4. Suivi des états des demandes**

* **Acteurs** : Gestionnaire de projet, Collaborateur Software Testing.
* **Étapes** :
  1. L'utilisateur s'authentifie.
  2. Accède à la section "Mes demandes" via le tableau de bord.
  3. Consulte la liste des demandes avec leur statut (en attente, en cours, traité, archivé).
  4. Applique des filtres (par date, priorité, statut) pour affiner la recherche.
  5. Clique sur une demande spécifique pour afficher les détails (historique, commentaires, état actuel).
* **Résultat attendu** : L'utilisateur visualise le statut de chaque demande en temps réel.

**5. Génération de rapports analytiques**

* **Acteurs** : Collaborateur Software Testing.
* **Étapes** :
  1. L'utilisateur s'authentifie.
  2. Accède à la section "Tableau de bord".
  3. Sélectionne une période ou un type de métrique (nombre de demandes, délais moyens de traitement, taux de succès des tests).
  4. Génère un rapport (PDF ou Excel) à partir des données sélectionnées.
  5. Télécharge le rapport ou l'envoie par email directement depuis la plateforme.
* **Résultat attendu** : Rapport généré et disponible au format souhaité.

**6. Notification des mises à jour**

* **Acteurs** : Gestionnaire de projet, Collaborateur Software Testing.
* **Étapes** :
  1. L'utilisateur reçoit une notification (email ou notification interne) lorsqu'une mise à jour critique se produit (changement d'état d'une demande, ajout d'un nouveau document, etc.).
  2. Clique sur la notification pour accéder directement à l'élément concerné.
  3. Consulte les détails de la mise à jour ou prend une action si nécessaire (par exemple, répondre à un commentaire).
* **Résultat attendu** : L'utilisateur est immédiatement informé des changements importants.

**7. Gestion des utilisateurs et des rôles**

* **Acteurs** : Administrateur.
* **Étapes** :
  1. L'administrateur s'authentifie avec ses identifiants.
  2. Accède à la section "Gestion des utilisateurs".
  3. Visualise la liste des utilisateurs existants et leurs rôles actuels.
  4. Ajoute un nouvel utilisateur en remplissant un formulaire (nom, email, rôle).
  5. Modifie ou révoque les permissions d'un utilisateur existant.
  6. Valide les changements et reçoit une confirmation.
* **Résultat attendu** : Les utilisateurs et leurs rôles sont mis à jour dans le système.

**8. Soumission de feedback**

* **Acteurs** : Tous les utilisateurs.
* **Étapes** :
  1. L'utilisateur s'authentifie.
  2. Accède à la section "Feedback".
  3. Remplit un formulaire avec ses commentaires ou suggestions (titre, description).
  4. Soumet le formulaire et reçoit une confirmation.
  5. L'équipe Software Testing traite le feedback et fournit un suivi (si nécessaire).
* **Résultat attendu** : Le feedback est enregistré et traité par l'équipe.

**9. Archivage des demandes clôturées**

* **Acteurs** : Collaborateur Software Testing.
* **Étapes** :
  1. L'utilisateur s'authentifie.
  2. Accède à la section "Archivage" via le tableau de bord.
  3. Consulte la liste des demandes clôturées prêtes à être archivées.
  4. Sélectionne les demandes à archiver et valide l'action.
* **Résultat attendu** : Les demandes sont archivées et restent consultables dans la section "Archivées".

**1. Création d'une demande**

* **Acteurs** : Gestionnaire de projet.
* **Étapes** :
  1. L'utilisateur s'authentifie.
  2. Accède à la section "Soumettre une demande" via le menu.
  3. Remplit le formulaire de demande (titre, description, priorité, contexte, deadline).
  4. Ajoute des fichiers annexes (par exemple, cahier des charges).
  5. Soumet la demande et reçoit une notification confirmant la soumission.
* **Résultat attendu** : La demande est enregistrée dans la base de données et visible dans la liste des demandes.

**2. Suivi des demandes**

* **Acteurs** : Gestionnaire de projet, Collaborateur Software Testing.
* **Étapes** :
  1. L'utilisateur s'authentifie.
  2. Accède à la section "Mes demandes" pour visualiser l'état des demandes.\n 3. Applique des filtres (date, statut, priorité) pour affiner la recherche.
  3. Clique sur une demande spécifique pour consulter ses détails (historique, commentaires, fichiers annexes).
  4. Peut ajouter des commentaires ou demander une mise à jour de la demande.
* **Résultat attendu** : Les demandes et leurs états sont affichés en temps réel avec la possibilité d'interagir.

**3. Mise à jour de l'état d'une demande**

* **Acteurs** : Collaborateur Software Testing.
* **Étapes** :
  1. L'utilisateur s'authentifie.
  2. Accède à la section "Demandes assignées" pour visualiser les demandes en cours.
  3. Sélectionne une demande spécifique.
  4. Change son état (par exemple : "en attente" → "en cours" → "traitée").
  5. Ajoute un commentaire ou un rapport de traitement (si nécessaire).
  6. Valide les changements.
* **Résultat attendu** : L'état de la demande est mis à jour et les parties concernées reçoivent une notification.

**4. Consultation des demandes archivées**

* **Acteurs** : Gestionnaire de projet, Collaborateur Software Testing.
* **Étapes** :
  1. L'utilisateur s'authentifie.
  2. Accède à la section "Demandes archivées" via le tableau de bord.
  3. Recherche une demande spécifique en utilisant des filtres (date, domaine, statut).
  4. Consulte les détails de la demande (historique, fichiers, commentaires).
* **Résultat attendu** : Les informations sur les demandes clôturées sont consultables et bien organisées.

**5. Réaffectation d'une demande**

* **Acteurs** : Collaborateur Software Testing, Administrateur.
* **Étapes** :
  1. L'utilisateur s'authentifie.
  2. Accède à la section "Gestion des demandes".
  3. Recherche une demande assignée à un utilisateur spécifique.
  4. Modifie l'utilisateur assigné (par exemple : transférer la demande à un autre collaborateur).
  5. Valide les changements.
* **Résultat attendu** : La demande est réaffectée à un autre collaborateur et les parties concernées reçoivent une notification.

**6. Clôture d'une demande**

* **Acteurs** : Collaborateur SoftwareTesting.
* **Étapes** :
  1. L'utilisateur s'authentifie.
  2. Accède à la section "Demandes en cours".
  3. Sélectionne une demande terminée.
  4. Ajoute un rapport de clôture (description des actions réalisées, résolution).
  5. Change le statut de la demande en "clôturée".
  6. Valide l’action.
* **Résultat attendu** : La demande est clôturée et transférée dans la section "Archivées".

**7. Suppression d'une demande**

* **Acteurs** : Gestionnaire de projet, Administrateur.
* **Étapes** :
  1. L'utilisateur s'authentifie avec les permissions nécessaires.
  2. Accède à la section "Gestion des demandes".
  3. Recherche une demande spécifique.
  4. Supprime la demande en confirmant l’action.
  5. Un message de confirmation s’affiche et la demande est retirée du système.
* **Résultat attendu** : La demande est supprimée définitivement.

**8. Ajout de commentaires ou pièces jointes**

* **Acteurs** : Gestionnaire de projet, Collaborateur Software Testing.
* **Étapes** :
  1. L'utilisateur s'authentifie.
  2. Accède à la section "Mes demandes" ou "Demandes assignées".
  3. Sélectionne une demande spécifique.
  4. Ajoute un commentaire décrivant une mise à jour ou une clarification.
  5. Télécharge des fichiers supplémentaires si nécessaire (documents, captures d’écran).
  6. Valide les modifications.
* **Résultat attendu** : Les commentaires et fichiers sont ajoutés à la demande et visibles par les parties concernées.

### \*\*Diagramme de classes complet :\*\*

---

### \*\*1. Chatbot\*\*

- \*\*Attributs :\*\*

- `id : Long` : Identifiant unique du chatbot.

- `historique : String` : Historique des conversations.

- \*\*Méthodes :\*\*

- `générerRéponse(question : String) : String` : Génère une réponse à une question posée par l'utilisateur.

- \*\*Relations :\*\*

- Intègre plusieurs \*\*Conversation\*\*.

---

### \*\*2. Utilisateur\*\*

- \*\*Attributs :\*\*

- `id : String` : Identifiant unique de l'utilisateur.

- `nom : String` : Nom de l'utilisateur.

- `prénom : String` : Prénom de l'utilisateur.

- `email : String` : Adresse e-mail.

- `rôle : String` : Rôle de l'utilisateur (utilisateur, admin, expert).

- \*\*Méthodes :\*\*

- `getDemandes() : List<Demande>` : Récupère les demandes soumises.

- `consulterDocuments() : List<Document>` : Liste les documents consultés.

- `recevoirNotification(notification : Notification) : void` : Reçoit une notification.

- \*\*Relations :\*\*

- Soumet des \*\*Demandes\*\*.

- Consulte des \*\*Documents\*\*.

- Reçoit des \*\*Notifications\*\*.

- Participe à des \*\*Conversations\*\*.

---

### \*\*3. Demande\*\*

- \*\*Attributs :\*\*

- `identifiant : String` : Identifiant unique de la demande.

- `titre : String` : Titre de la demande.

- `description : String` : Description de la demande.

- `priorité : PrioritéDemande` : Niveau de priorité.

- `statut : StatutDemande` : État de traitement de la demande.

- `duréeEstimée : Integer` : Durée estimée de traitement (en jours).

- `dateSoumission : LocalDate` : Date de soumission de la demande.

- \*\*Méthodes :\*\*

- `getNotifications() : List<Notification>` : Liste des notifications associées.

- `mettreÀJourStatut(statut : StatutDemande) : void` : Modifie le statut.

- \*\*Relations :\*\*

- Soumise par \*\*Utilisateur\*\*.

- Génère des \*\*Notifications\*\*.

- Influence les \*\*Statistiques\*\*.

---

### \*\*4. Document\*\*

- \*\*Attributs :\*\*

- `id : Long` : Identifiant du document.

- `titre : String` : Titre du document.

- `contenu : String` : Contenu textuel du document.

- `catégorie : String` : Catégorie (guide, tutoriel, politique, etc.).

- `dateAjout : LocalDate` : Date d'ajout.

- \*\*Méthodes :\*\*

- `ajouterDocument() : void` : Ajoute un nouveau document.

- `mettreÀJourContenu(contenu : String) : void` : Met à jour le contenu.

- \*\*Relations :\*\*

- Consulté par \*\*Utilisateur\*\*.

- Peut être référencé dans des \*\*Notifications\*\*.

---

### \*\*5. Notification\*\*

- \*\*Attributs :\*\*

- `id : Long` : Identifiant unique de la notification.

- `message : String` : Contenu du message.

- `dateEnvoi : LocalDateTime` : Date et heure d'envoi.

- \*\*Méthodes :\*\*

- `marquerCommeLue() : void` : Marque la notification comme lue.

- \*\*Relations :\*\*

- Reliée à une \*\*Demande\*\*.

- Reçue par \*\*Utilisateur\*\*.

---

### \*\*6. Conversation\*\*

- \*\*Attributs :\*\*

- `id : Long` : Identifiant de la conversation.

- `messages : List<String>` : Liste des messages échangés.

- `dateDébut : LocalDateTime` : Date de début de la conversation.

- `dateFin : LocalDateTime` : Date de fin (si terminée).

- \*\*Méthodes :\*\*

- `ajouterMessage(message : String) : void` : Ajoute un message.

- `terminerConversation() : void` : Met fin à la conversation.

- \*\*Relations :\*\*

- Reliée à \*\*Chatbot\*\*.

- Associée à \*\*Utilisateur\*\*.

---

### \*\*7. Statistique\*\*

- \*\*Attributs :\*\*

- `id : String` : Identifiant.

- `nbDemandesTraitées : Integer` : Nombre de demandes traitées.

- `nbDemandesEnCours : Integer` : Nombre de demandes en cours.

- `délaiMoyenTraitement : Double` : Temps moyen de traitement (en jours).

- \*\*Méthodes :\*\*

- `calculerDélaiMoyen() : Double` : Calcule le délai moyen.

- `mettreÀJourStatistiques() : void` : Met à jour les indicateurs.

---

### \*\*8. Énumérations :\*\*

- \*\*StatutDemande :\*\*

- `EN\_ATTENTE`

- `EN\_COURS`

- `TRAITÉE`

- \*\*PrioritéDemande :\*\*

- `FAIBLE`

- `MOYENNE`

- `ÉLEVÉE`

**9. BaseDeConnaissance**

* **Attributs :**
  + id : Long : Identifiant.
  + description : String : Description.
* **Méthodes :**
  + rechercherArticle(motCle : String) : List<Article> : Recherche des articles par mot-clé.
* **Relations :**
  + Contient des **Articles**.
  + Consultée par le **Chatbot**.

**10. Bibliothèque *(Nouvelle classe ajoutée)***

* **Attributs :**
  + id : Long : Identifiant.
  + nom : String : Nom de la bibliothèque.
  + description : String : Description générale.
* **Méthodes :**
  + ajouterArticle(article : Article) : void : Ajoute un article.
  + supprimerArticle(articleId : Long) : void : Supprime un article.
  + listerArticles() : List<Article> : Liste tous les articles.
* **Relations :**
  + Contient plusieurs **Articles**.
  + Consultée par des **Utilisateurs**.

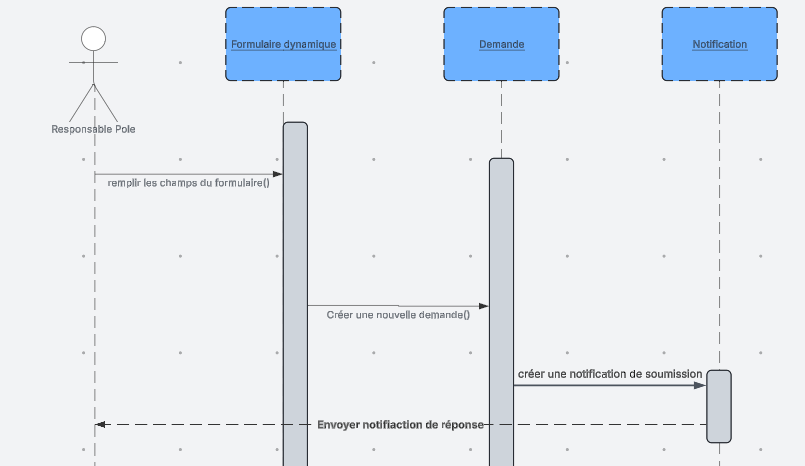
**11. Article *(Nouvelle classe ajoutée)***

* **Attributs :**
  + id : Long : Identifiant de l'article.
  + titre : String : Titre de l'article.
  + contenu : String : Contenu détaillé.
  + auteur : String : Nom de l'auteur.
  + datePublication : LocalDate : Date de publication.
  + motsCles : List<String> : Liste des mots-clés.
* **Méthodes :**
  + mettreÀJourContenu(contenu : String) : void : Met à jour le contenu.
  + ajouterMotCle(mot : String) : void : Ajoute un mot-clé.
* **Relations :**
  + Fait partie de la **Bibliothèque**.
  + Référencé par la **BaseDeConnaissance**.
  + Consulté par des **Utilisateurs**.

### ****Réalisation****

* **Attributs :**
  + id : Long : Identifiant.
  + titre : String : Titre de la réalisation.
  + description : String : Description détaillée.
  + dateDébut : LocalDate : Date de début.
  + dateFin : LocalDate : Date de fin.
  + statut : String : État d'avancement (En cours, Terminée, Annulée).
* **Méthodes :**
  + ajouterRéalisation() : void : Ajoute une nouvelle réalisation.
  + mettreÀJourStatut(nouveauStatut : String) : void : Met à jour le statut.
  + ajouterDocument(document : Document) : void : Associe un document.
* **Relations :**
  + Réalisée par des **Utilisateurs**.
  + Issue d'une **Demande**.
  + Peut contenir des **Documents** et des **Articles** pour référence.

Diagrammes de séquence :

* Création des demandes :
* Suivi d’une demande

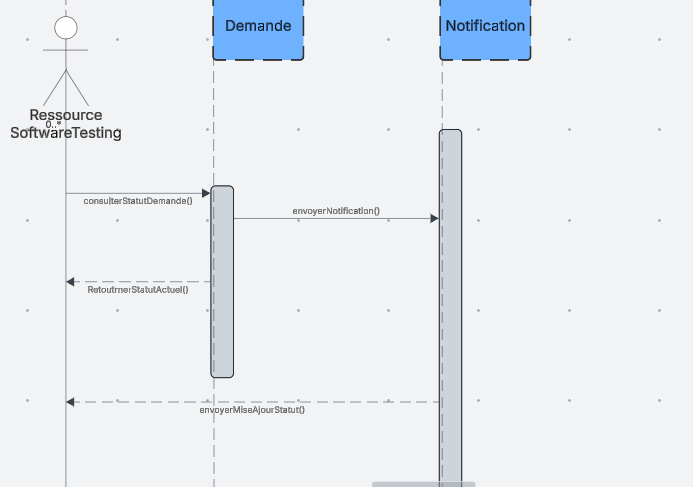


Diagramme d’activités :

Début

└── Responsable Pôle : Remplir le formulaire

├─ Vérification des champs obligatoires

│ ├─ [Oui] → Soumission de la demande

│ └─ [Non] → Message d’erreur → Retour à la saisie du formulaire

└─ Création d'une nouvelle demande

└─ Création de la notification de soumission

└─ Envoi de la notification au Responsable Pôle

└─ Responsable Pôle : Recevoir la notification de soumission

Ressource Software Testing : S’authentifier

├─ Vérification des identifiants

│ ├─ [Valide] → Accéder aux demandes assignées

│ │ └─ Récupérer les détails de la demande

│ │ └─ Afficher les détails au testeur

│ │ └─ Modifier le statut de la demande et/ou ajouter un rapport

│ │ └─ Créer et envoyer la notification de mise à jour

│ │ └─ Responsable Pôle : Recevoir la notification de mise à jour

│ └─ [Invalide] → Afficher un message d’erreur → Retour à l'authentification

Dashboard

├─ Accéder au dashboard

│ ├─ Visualiser la liste des demandes

│ ├─ Consulter les statistiques de traitement

│ └─ Filtrer et rechercher des demandes

└─ Retour au menu principal

Chatbot

├─ Lancer le chatbot

│ ├─ Poser une question ou demander de l'aide

│ │ ├─ [Réponse suffisante] → Fin de l’interaction chatbot

│ │ └─ [Réponse insuffisante] → Redirection vers le chat direct

└─ Fin de l’interaction chatbot (si réponse suffisante)

Chat direct

├─ Contacter un agent

│ ├─ Envoyer et recevoir des messages en temps réel

│ └─ Clôturer la conversation

└─ Fin de l’interaction chat direct

Fin

## **1. Gestion des demandes**

### ****Acteurs :****

* Gestionnaire de projet (acteur principal)
* Système (sous-système de gestion des demandes)
* Ressource SoftwareTesting (acteur secondaire)

### ****Description du diagramme :****

Ce diagramme illustre le processus de soumission et de suivi d'une demande :

* **Rectangle d'acteur (stickman)** : Représente le Gestionnaire de projet et la Ressource SoftwareTesting.
* **Boîtes de vie (lifelines)** : Gestionnaire de projet, Interface Utilisateur, Système, Ressource SoftwareTesting.
* **Messages (flèches)** : Représentent les appels de méthodes, notifications et retours de données.
* **Cadres d'interaction (alt, opt)** : Décrivent les conditions comme la validation ou l’état de la demande.

### ****Séquence :****

1. **Gestionnaire de projet → Interface Utilisateur** : Remplit et soumet le formulaire dynamique. (Flèche pleine avec "submitRequest()")
2. **Interface Utilisateur → Système** : Envoie les données pour validation. (Flèche pleine avec "validateRequest()")
3. **Système → Interface Utilisateur** : Retour de validation positive ou message d'erreur. (Cadre alt pour les deux possibilités)
4. **Système → Gestionnaire de projet** : Notification de soumission réussie. (Flèche en pointillés "notifySubmission()")
5. **Ressource SoftwareTesting → Système** : Consulte et met à jour l'état de la demande. (Flèche pleine "updateRequestStatus()")
6. **Système → Gestionnaire de projet** : Notification du changement d’état. (Flèche en pointillés "notifyStatusUpdate()")

## **2. Bibliothèque documentaire**

### ****Acteurs :****

* Collaborateur externe (acteur principal)
* Système (moteur de recherche documentaire)

### ****Description du diagramme :****

Ce diagramme illustre l’accès et la recherche dans la bibliothèque documentaire :

* **Stickman** : Collaborateur externe.
* **Lifelines** : Collaborateur externe, Interface Utilisateur, Moteur de recherche documentaire.
* **Flèches pleines** : Actions de consultation et recherche.
* **Flèches en pointillés** : Résultats renvoyés.
* **Cadre opt** : Pour la consultation d’un document spécifique.

### ****Séquence :****

1. **Collaborateur externe → Interface Utilisateur** : Lance la fonction de recherche. ("searchDocuments(filters)")
2. **Interface Utilisateur → Moteur de recherche documentaire** : Requête avec filtres appliqués.
3. **Moteur de recherche documentaire → Interface Utilisateur** : Renvoie les résultats. (Flèche en pointillés)
4. **Collaborateur externe → Interface Utilisateur** : Sélectionne un document pour consultation. (Cadre opt)
5. **Interface Utilisateur → Moteur de recherche documentaire** : Demande de consultation.
6. **Moteur de recherche documentaire → Collaborateur externe** : Affiche le contenu du document.

## **3. Chatbot interactif**

### ****Acteurs :****

* Utilisateur (Gestionnaire, Ressource ou Collaborateur)
* Chatbot
* Expert (si escalade nécessaire)

### ****Description du diagramme :****

Ce diagramme illustre l’interaction utilisateur-chatbot et l’escalade vers un expert :

* **Stickman** : Utilisateur.
* **Lifelines** : Utilisateur, Chatbot, Base de données documentaire, Expert.
* **Cadres alt** : Gèrent la distinction entre question simple et complexe.
* **Flèches en pointillés** : Réponses ou suggestions du Chatbot.

### ****Séquence :****

1. **Utilisateur → Chatbot** : Pose une question. ("askQuestion()")
2. **Chatbot → Base de données documentaire** : Recherche de réponse.
3. **Cadre alt** :
   * **Si réponse trouvée :**
     + **Base de données → Chatbot** : Renvoie la réponse ou un document pertinent.
     + **Chatbot → Utilisateur** : Fournit la réponse. (Flèche en pointillés)
   * **Sinon (escalade) :**
     + **Chatbot → Expert** : Transmet la question complexe.
     + **Expert → Chatbot** : Fournit une réponse technique.
     + **Chatbot → Utilisateur** : Transmet la réponse.

## **4. Tableau de bord analytique**

### ****Acteurs :****

* Gestionnaire de projet (acteur principal)
* Système (module analytique)

### ****Description du diagramme :****

Ce diagramme montre la consultation des indicateurs de performance et l’exportation des données :

* **Stickman** : Gestionnaire de projet.
* **Lifelines** : Gestionnaire de projet, Interface Utilisateur, Module analytique.
* **Cadre opt** : Pour l’option d’exportation des données.

### ****Séquence :****

1. **Gestionnaire de projet → Interface Utilisateur** : Accède au tableau de bord. ("viewDashboard()")
2. **Interface Utilisateur → Module analytique** : Demande des données de performance.
3. **Module analytique → Interface Utilisateur** : Affiche les KPI et graphiques. (Flèche en pointillés)
4. **Cadre opt (Exportation) :**
   * **Gestionnaire → Interface Utilisateur** : Clique sur "Exporter en Excel".
   * **Interface → Module analytique** : Génère le fichier.
   * **Module → Gestionnaire** : Fournit le fichier Excel téléchargé.

### Présentation de mon sujet de stage PFE devant le manager

---

#### \*\*Introduction\*\*

Bonjour, je m'appelle \*\*Hanane AMTDI\*\*, étudiante en dernière année d’ingénierie informatique à l’ENSA Fès. Je réalise mon \*\*Projet de Fin d’Études (PFE)\*\* au sein de l’équipe \*\*Software Testing\*\*, sous la supervision de mon encadrante et en collaboration avec des membres expérimentés de l’équipe.

---

#### \*\*Présentation du sujet\*\*

Mon projet porte sur la \*\*mise en place d’un portail web Software Testing\*\* ayant pour objectif la \*\*centralisation et la gestion des demandes\*\*. Actuellement, l’entité fait face à des problématiques telles que la dispersion des informations, l'absence de processus standardisé pour soumettre et suivre les demandes, et un manque d’outils de reporting. Le portail que je développe vise à répondre à ces besoins en offrant une plateforme intuitive et complète pour :

- La \*\*soumission et le suivi des demandes\*\* avec notifications automatiques.

- L'accès à une \*\*bibliothèque documentaire\*\* riche et bien structurée.

- Un \*\*chatbot interactif\*\* permettant de fournir des réponses aux questions techniques fréquentes et des suggestions de documents pertinents.

- Un \*\*tableau de bord analytique\*\* pour visualiser les KPI, générer des graphiques interactifs et exporter les données.

---

#### \*\*Intégration de l’intelligence artificielle\*\*

Pour enrichir l’expérience utilisateur et automatiser les réponses, le projet intègre un \*\*chatbot intelligent\*\* reposant sur des modèles de traitement du langage naturel (NLP) permettant :

- D’apporter des \*\*réponses automatiques\*\* aux questions fréquentes.

- De \*\*suggérer des documents pertinents\*\* grâce à des algorithmes de recherche intelligente.

- D’\*\*escalader les questions complexes\*\* vers un expert lorsque nécessaire.

Cette intégration d’IA vise à améliorer la réactivité du système et à optimiser la gestion des connaissances.

---

#### \*\*Technologies et outils utilisés\*\*

Pour répondre aux besoins fonctionnels et non fonctionnels du projet, j’utilise une stack technologique moderne et performante :

- \*\*Backend\*\* :

- \*\*Spring Boot\*\* : Gestion des API REST et de la logique métier.

- \*\*FastAPI\*\* : Pour le service du chatbot et les interactions avec l'IA.

- \*\*Redis\*\* : Gestion des sessions et des notifications en temps réel.

- \*\*Kafka\*\* : Gestion de la communication asynchrone entre services (microservices).

- \*\*PostgreSQL\*\* : Gestion de la base de données relationnelle.

- \*\*Frontend\*\* :

- \*\*React.js\*\* : Création d’une interface utilisateur réactive et ergonomique.

- \*\*Autres outils\*\* :

- \*\*Docker\*\* : Conteneurisation et déploiement des services.

- \*\*Git\*\* : Gestion de version et collaboration.

- \*\*Excel\*\* : Exportation des rapports analytiques.

---

#### \*\*Conclusion\*\*

Ce projet vise à améliorer considérablement la \*\*gestion des demandes\*\*, la \*\*centralisation des informations\*\* et l’\*\*accessibilité aux connaissances sur le testing\*\*. Grâce à l'intégration de l’intelligence artificielle et de technologies modernes, le portail offrira une solution complète répondant aux attentes des différentes parties prenantes. Je suis ravie de collaborer avec l’équipe \*\*Software Testing\*\* et de contribuer activement à ce projet stratégique.