1. Diagramme de Cas d'Utilisation

Explication :

- **But :** Illustrer les interactions entre les acteurs (par exemple, Étudiant et Administrateur) et le système.
- **Ce qu'il montre :**
    - Les actions principales que chaque acteur peut réaliser (comme se connecter, consulter, télécharger ou ajouter des documents).
    - Les relations entre ces actions, par exemple via les stéréotypes <<include>> et <<extend>> pour clarifier les dépendances et les comportements communs.
- **Pourquoi c'est utile :**
    - Il permet de comprendre rapidement les fonctionnalités principales du système du point de vue des utilisateurs.
    - Il sert de base pour définir les exigences fonctionnelles.

---

2. Diagramme de Classes

Explication :

- **But :** Définir la structure statique de l'application en identifiant les classes principales, leurs attributs et leurs méthodes ainsi que les relations entre elles (héritage, associations, agrégations, etc.).
- **Ce qu'il montre :**
    - Les classes comme *Utilisateur* (avec ses spécialisations *Étudiant* et *Administrateur*), *Document*, *Catégorie*, etc.
    - Les relations entre ces classes, par exemple : un étudiant peut consulter plusieurs documents, un document appartient à une catégorie, etc.
- **Pourquoi c'est utile :**
    - Il sert de plan pour la conception du système, assurant que toutes les entités et leurs interactions sont bien définies.
    - Il facilite la compréhension de la structure du système pour les développeurs et parties prenantes.

---

3. Diagramme de Séquence

Explication :

- **But :** Décrire le comportement dynamique du système en représentant l'ordre chronologique des interactions entre les objets lors de l'exécution d'une fonctionnalité.
- **Ce qu'il montre :**

    - Par exemple, le scénario où un étudiant se connecte et télécharge un document ou celui où un administrateur ajoute un document.
    - Les échanges de messages entre les acteurs (Étudiant, Administrateur) et le Système, en mettant en évidence l'ordre et la logique des appels de méthodes.
- **Pourquoi c'est utile :**
    - Il permet d'identifier les étapes clés dans la réalisation d'une fonctionnalité.
    - Il aide à détecter des problèmes de conception en simulant le comportement du système au moment de l'exécution.

---

4. Diagramme d'État-Transition (ou State Machine Diagram / Statechart Diagram)

Explication :

- **But :** Représenter le comportement d'un objet ou d'un système en détaillant les états successifs qu'il peut adopter et les événements qui provoquent la transition d'un état à l'autre.
- **Ce qu'il montre :**
    - Par exemple, pour le cycle de vie d'un document : depuis sa création, sa validation, sa consultation jusqu'à sa suppression ou son archivage.
    - Les événements ou actions qui déclenchent les changements d'état (comme "ValiderDocument()" ou "Supprimer()").
- **Pourquoi c'est utile :**
    - Il fournit une vision claire du flux d'état d'un élément critique du système.
    - Il aide à vérifier que toutes les transitions nécessaires sont bien prises en compte dans la conception.

---

5. Diagramme d'Objet

Explication :

- **But :** Fournir un instantané (une « photo ») de l'état du système à un moment donné en montrant les instances concrètes des classes et leurs relations.
- **Ce qu'il montre :**
    - Des objets comme un étudiant avec ses attributs réels (ex : nom, email) et un document avec ses valeurs (titre, statut).
    - Les liens entre ces objets, par exemple un étudiant consulte un document ou un administrateur a ajouté un document.
- **Pourquoi c'est utile :**

- Il permet de vérifier la cohérence de la modélisation des classes en montrant des exemples concrets d'instances.
- Il aide à illustrer comment le système est utilisé à un instant T, ce qui peut être très parlant pour comprendre les relations pratiques.

---

6. Diagramme de Déploiement

Explication :

- **But :** Montrer l'architecture physique du système, c'est-à-dire comment les différents composants logiciels sont déployés sur des matériels ou nœuds (serveurs, clients, bases de données).
- **Ce qu'il montre :**
  - Les nœuds physiques ou virtuels (par exemple, le client Web, le serveur Web et le serveur de base de données).
  - Les artifacts déployés sur chaque nœud (ex : Application Django sur le serveur Web, base MySQL sur le serveur de base de données).
  - Les connexions entre ces nœuds (flux de données, requêtes HTTP ou SQL).
- **Pourquoi c'est utile :**
  - Il donne une vision globale de l'infrastructure technique.
  - Il permet de comprendre comment les différents composants interagissent sur le plan physique, ce qui est essentiel pour la mise en production.

---

7. Diagramme de Package

Explication :

- **But :** Organiser et structurer le modèle UML en regroupant les classes, diagrammes et autres éléments connexes en packages.
- **Ce qu'il montre :**
  - Les différents modules ou domaines fonctionnels du système (ex : Gestion des Documents, Gestion des Utilisateurs).
  - Les dépendances ou interactions entre ces packages.
- **Pourquoi c'est utile :**
  - Il facilite la compréhension et la maintenance de grandes architectures en segmentant le système en parties logiques.
  - Il permet de voir d’un seul coup d’œil comment les modules de ton application sont interconnectés.