I understand you're looking for detailed diagrams representing the system for handling requests, including approval remarks from both the **HOD** and **Director**. Based on the flow you mentioned, the system involves multiple stages of approval and remarking. Here’s how you could approach the system's ER diagram, class diagram, and UML diagram, which represent the core components of your system.

**1. ER Diagram (Entity-Relationship Diagram)**

This diagram focuses on representing the entities involved in the system and how they interact with one another. Below are the entities and their relationships:

**Entities:**

* **Request**
  + Attributes: requestID, requestDate, status, finalStatus
* **User (HOD, Director)**
  + Attributes: userID, name, role
* **Remark**
  + Attributes: remarkID, text, remarkDate, requestID (Foreign Key), userID (Foreign Key)
* **Approval**
  + Attributes: approvalID, approvalStatus, approvalDate, requestID (Foreign Key), userID (Foreign Key)

**Relationships:**

* **Request** to **Remark**: One request can have many remarks (from HOD, Director).
* **User** to **Remark**: A user (HOD or Director) can create multiple remarks.
* **Request** to **Approval**: Each request goes through multiple approval stages (HOD approval and Director approval).
* **User** to **Approval**: A user can approve a request.

**ER Diagram Example:**

pgsql

Copy

+-------------------+ +-------------------+ +-------------------+

| Request | | Remark | | User |

+-------------------+ +-------------------+ +-------------------+

| requestID (PK) |--------| remarkID (PK) |--------| userID (PK) |

| requestDate | | text | | name |

| status | | remarkDate | | role (HOD, Director)|

| finalStatus | | requestID (FK) | +-------------------+

+-------------------+ | userID (FK) |

+-------------------+

|

|

+-------------------+

| Approval |

+-------------------+

| approvalID (PK) |

| approvalStatus |

| approvalDate |

| requestID (FK) |

| userID (FK) |

+-------------------+

**2. Class Diagram**

The class diagram represents the system's core structure, with classes (like Request, User, Remark, and Approval) and their relationships.

**Classes:**

1. **Request**
   * Attributes: requestID, requestDate, status, finalStatus
   * Methods: addRemark(), updateStatus(), getRequestStatus()
2. **User (Abstract Class for HOD and Director)**
   * Attributes: userID, name, role
   * Methods: addRemark(), approveRequest()
3. **HOD (Subclass of User)**
   * Attributes: Inherited from User
   * Methods: approveRequest(), addHODRemark()
4. **Director (Subclass of User)**
   * Attributes: Inherited from User
   * Methods: approveRequest(), addDirectorRemark()
5. **Remark**
   * Attributes: remarkID, text, remarkDate, requestID, userID
   * Methods: displayRemark()
6. **Approval**
   * Attributes: approvalID, approvalStatus, approvalDate, requestID, userID
   * Methods: updateApprovalStatus()

**Class Diagram Example:**

pgsql

Copy

+-------------------+ +---------------------+

| Request | | User |

+-------------------+ +---------------------+

| - requestID |<>----| - userID |

| - requestDate | | - name |

| - status | | - role |

| - finalStatus | +---------------------+

| | ^ ^

+-------------------+ | |

| | |

| | |

| | |

+-------------------+ +--------------------+

| Remark | | HOD |

+-------------------+ +--------------------+

| - remarkID | | - approveRequest() |

| - text | | - addHODRemark() |

| - remarkDate | +--------------------+

| - userID |

| - requestID |

+-------------------+

^

|

|

+---------------------+

| Director |

+---------------------+

| - approveRequest() |

| - addDirectorRemark()|

+---------------------+

**3. UML Use Case Diagram**

The UML use case diagram shows the interaction between users (actors) and the system.

**Actors:**

* HOD
* Director
* System

**Use Cases:**

1. **HOD**:
   * Approve or reject request
   * Add remarks
   * View request status
2. **Director**:
   * Approve or reject request
   * Add remarks
   * View request status
3. **System**:
   * Log approval status
   * Notify HOD/Director when action is required
   * Finalize request status

**UML Use Case Diagram Example:**

pgsql

Copy

+-------------------------+

| <<System>> |

| |

| - Log Approval Status |

| - Notify HOD/Director |

| - Finalize Request |

+-------------------------+

^ ^

/ \ / \

/ \ / \

+--------+ +----------+ +--------+

| HOD | | Director | | System |

+--------+ +----------+ +--------+

| | |

+------------+ +------------+ +-----------+

| Approve/Reject| | Approve/Reject| | Finalize Request |

| Add Remarks | | Add Remarks | | Log and Notify |

| View Status | | View Status | +-------------------+

+----------------+ +----------------+

**Explanation of Flow in the System:**

1. **Pending Request**:
   * A **Request** is initially created by a user and submitted for approval.
2. **HOD's Remarks and Approval**:
   * The **HOD** reviews the request, adds their **remark**, and approves or rejects the request.
3. **Director's Review and Remarks**:
   * After HOD's review, the **Director** reviews the request, adds their **remark**, and approves or rejects the request.
4. **Final Approval in the DDNIT System**:
   * After the **Director** finalizes the approval, the system logs the status and sends notifications as required.

**Summary of Diagrams:**

1. **ER Diagram (Entity-Relationship Diagram)**: Defines entities like **Request**, **User**, **Remark**, and **Approval**, and their relationships.
2. **Class Diagram**: Describes the **Request**, **User**, **Remark**, and **Approval** classes, along with their attributes and methods.
3. **UML Use Case Diagram**: Represents the interaction between **HOD**, **Director**, and the **System** through actions like approval, remarking, and finalizing the request.

**Tools for Visualizing:**

To draw these diagrams visually, you can use:

* **Lucidchart** (easy online tool)
* **draw.io** (free and flexible tool)
* **Microsoft Visio**
* **StarUML** (for more technical UML diagrams)

Would you like further help with using any of these tools to generate the diagrams, or do you have additional details to include? Let me know how you'd like to proceed!