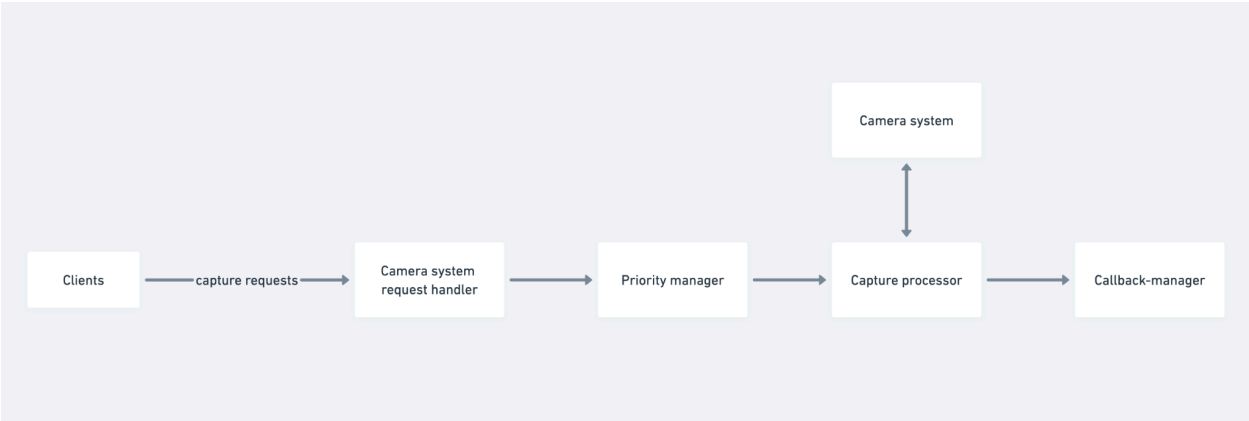
CAMERA SYSTEM DESIGN

High-Level Design (HLD)

Components and Responsibilities

1. **Request Handler**
    a. Receive and queue capture requests from clients.
    b. Pass requests to the Priority Manager for prioritization.
2. **Priority Manager**
    a. Determine the order of request processing based on urgency.
    b. Ensure high-urgency requests are processed with minimal delay.
3. **Capture Processor**
    a. Process capture requests in the order determined by the Priority Manager.
    b. Interact with the Camera System to capture images.
    c. Report results back to the Callback Manager.
4. **Camera System**
    - Interface with the camera hardware.
    - Start the capture process asynchronously.
    - Register success and failure callbacks.
    - Return capture results or error messages.
2. **Callback Manager**
    - Manage success and failure callbacks.
    - Invoke the appropriate callback once a capture request is completed.

High Level Design Diagram



Logical Flow for Concurrent Requests

1. **Request Submission**: Clients submit capture requests to the Request Handler, including urgency levels.
2. **Request Queuing**: The Request Handler queues the requests based on urgency using the Priority Manager.
3. **Processing Requests**: The Capture Processor processes the queued requests, prioritizing high-urgency requests.
4. **Image Capture**: The Capture Processor interacts with the Camera System to capture images.
5. **Callback Invocation**: Based on the result, the Callback Manager invokes either the success or failure callback.

Example :

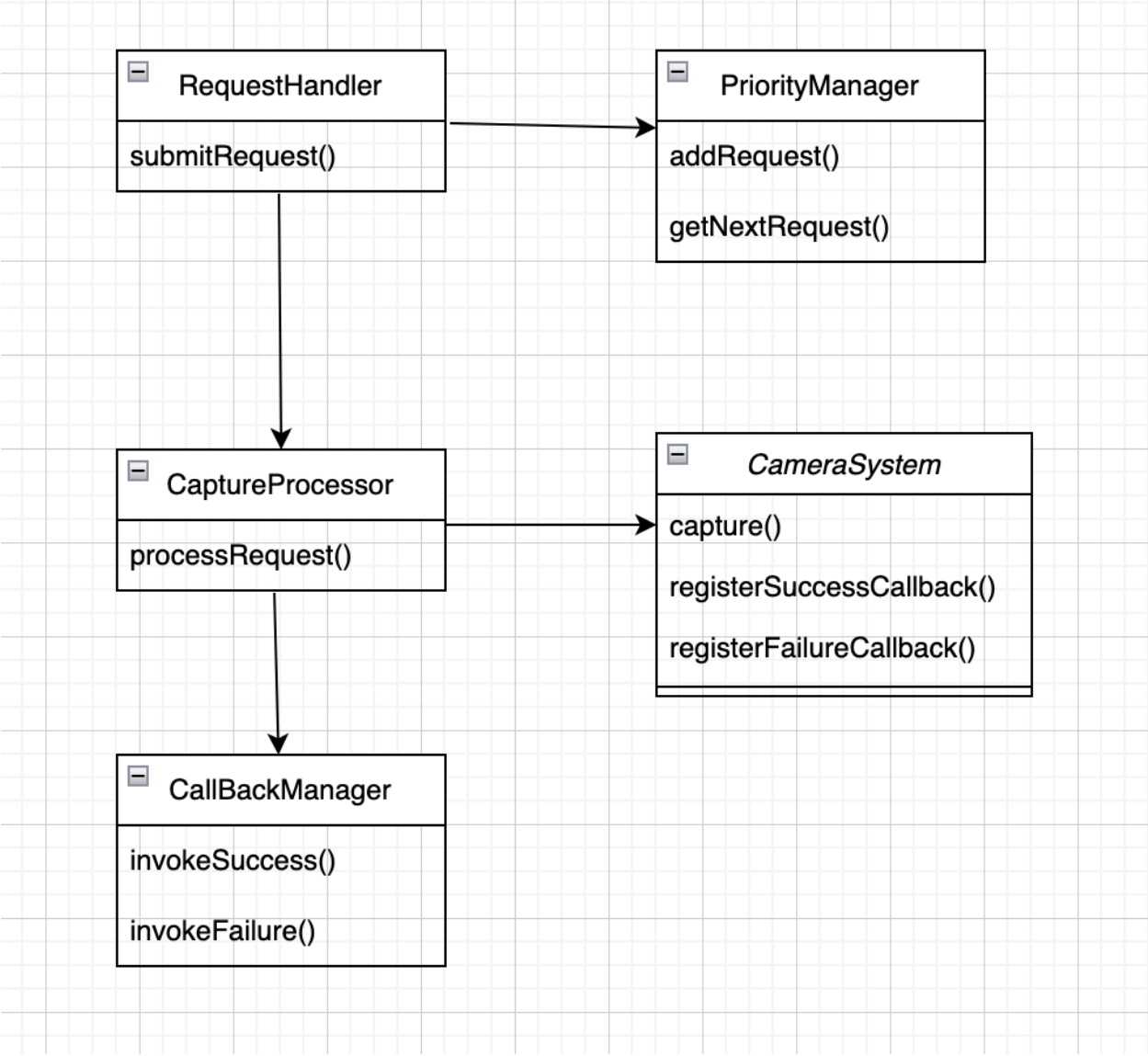
1. Client A submits a capture request with high urgency.
2. Client B submits a capture request with low urgency.
3. Both requests are received by the Request Handler.
4. The Priority Manager prioritizes Client A's request over Client B's.
5. The Capture Processor processes Client A's request first.
6. The Camera System captures the image for Client A.
7. The Callback Manager invokes Client A's success/failure callback.
8. The Capture Processor then processes Client B's request.
9. The Camera System captures the image for Client B.
10. The Callback Manager invokes Client B's success/failure callback.

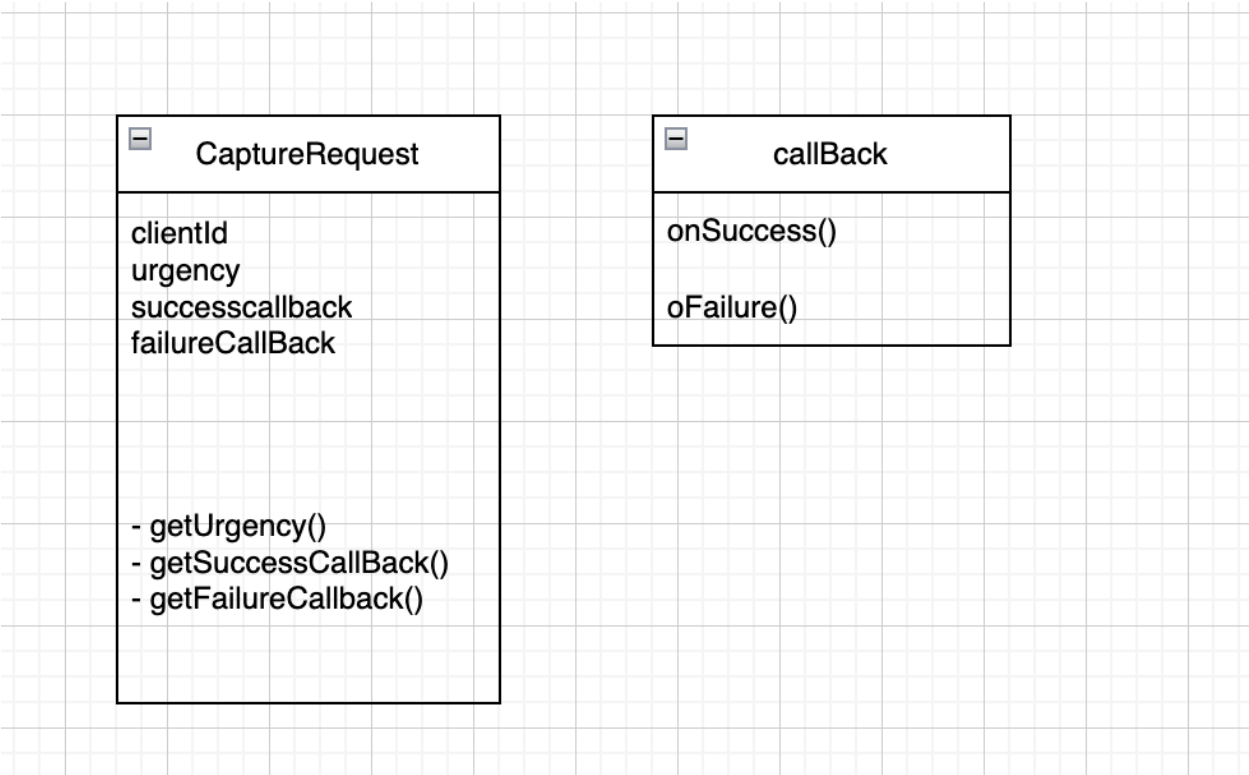
Low level design

- **CameraSystem**
  - **Methods**:
    - `submitCaptureRequest(request: CaptureRequest)`
    - `registerSuccessCallback(callback: Callable)`
    - `registerFailureCallback(callback: Callable)`
  - **Description**: Core class managing the camera system.
- **CaptureRequest**
  - **Attributes**:
  -
    - `urgency: int`
    - `successCallback: Callable`
    - `failureCallback: Callable`
  - **Description**: Represents a capture request with urgency and callbacks.
- **RequestHandler**
  - **Attributes**:
    - `priorityManager: PriorityManager`
  - **Methods**:
    - `addRequest(request: CaptureRequest)`
  - **Description**: Handles incoming requests and manages the request queue.
- **CaptureProcessor**
  - **Attributes**:
    - `cameraSystem: CameraSystem`
    - `callbackManager: CallbackManager`
  - **Methods**:
    - `processRequests()`
  - **Description**: Processes capture requests from the queue.
- **PriorityManager**
  - **Attributes**:
    - `requestQueue: PriorityQueue`
  - **Methods**:
    - `assignPriority(request: CaptureRequest)`
    - `getNextRequest()`
  - **Description**: Manages request priorities and fetches the next request to process.
- **CallbackManager**
  - **Attributes**:
    - `successCallbacks: List[Callable]`
    - `failureCallbacks: List[Callable]`
  - **Methods**:
    - `registerSuccessCallback(callback: Callable)`

        - `registerFailureCallback(callback: Callable)`
        - `invokeCallback(callbackType: str, data: Any)`
    - **Description**: Manages and invokes callbacks.

LLD Diagram :





Detailed Sequence of Events for Concurrent Requests with Different Urgency Levels

1. **Client1** submits a capture request with `urgency=5`.
2. **Client2** submits a capture request with `urgency=9`.
3. **RequestHandler** adds both requests to the queue using `addRequest(request: CaptureRequest)`.
4. **PriorityManager** assigns priorities to the requests and orders them in the `requestQueue`.
5. **CaptureProcessor** calls `processRequests()` to start processing requests from the queue.
6. **CaptureProcessor** fetches the next request from `PriorityManager.getNextRequest()`, which returns **Client2**'s request due to higher urgency.
7. **CaptureProcessor** processes **Client2**'s request by calling `CameraSystem.captureImage()`.

8. **CameraSystem** captures the image and returns it to **CaptureProcessor**.
9. **CallbackManager** invokes the success callback for **Client2** using `invokeCallback("success", image)`.
10. **CaptureProcessor** then processes **Client1**'s request by fetching it from `PriorityManager.getNextRequest()`.
11. **CameraSystem** captures the image and returns it to **CaptureProcessor**.
12. **CallbackManager** invokes the success callback for **Client1** using `invokeCallback("success", image)`.

Priority Queue Management

- **PriorityManager** uses a priority queue to manage the requests. The priority queue orders requests based on their urgency levels.
- When a new request is added, it is placed in the queue according to its urgency.
- The `getNextRequest()` method fetches the request with the highest urgency for processing.

Priority Queue Implementation

- **Binary Heap**: The priority queue can be implemented using a binary heap:
    - **Insertion**: O(log n) time complexity.
    - **Removal**: O(log n) time complexity.
    - **Peek**: O(1) time complexity.
- **Alternatives**: Other data structures like balanced binary search trees (e.g., AVL trees) or treemaps could also be used based on specific requirements and performance considerations.

Callback Handling

- **CallbackManager** maintains lists of success and failure callbacks.
- Callbacks are registered using `registerSuccessCallback(callback: Callable)` and `registerFailureCallback(callback: Callable)`.
- When an image is successfully captured, `invokeCallback("success", image)` is called to invoke the success callback with the captured image