High-Level Design of Camera System

Client Interface:

- submitCaptureRequest(): Submits a request to capture an image.
- setSuccessCallback(): Registers a callback for successful image capture.
- setFailureCallback(): Registers a callback for failed image capture.

Capture Request Manager:

- manageRequest(): Manages incoming capture requests.
- prioritizeRequests(): Prioritises requests based on urgency.
- dispatchRequest(): Dispatches requests to the Camera System.

Urgency Comparator:

- compare(): Compares the urgency of different requests.

Camera System:

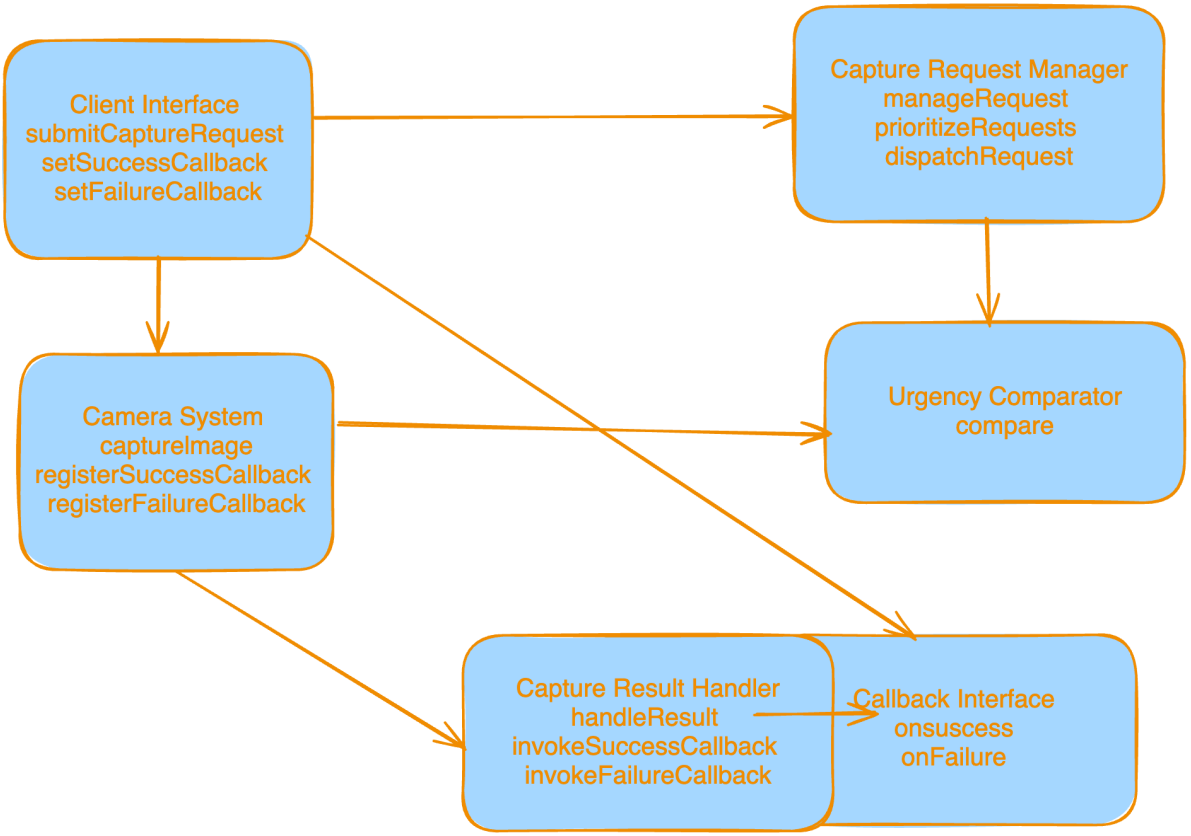
- captureImage(): Captures an image asynchronously.
- registerSuccessCallback(): Registers a callback for successful capture.
- registerFailureCallback(): Registers a callback for capture failure.

Capture Result Handler:

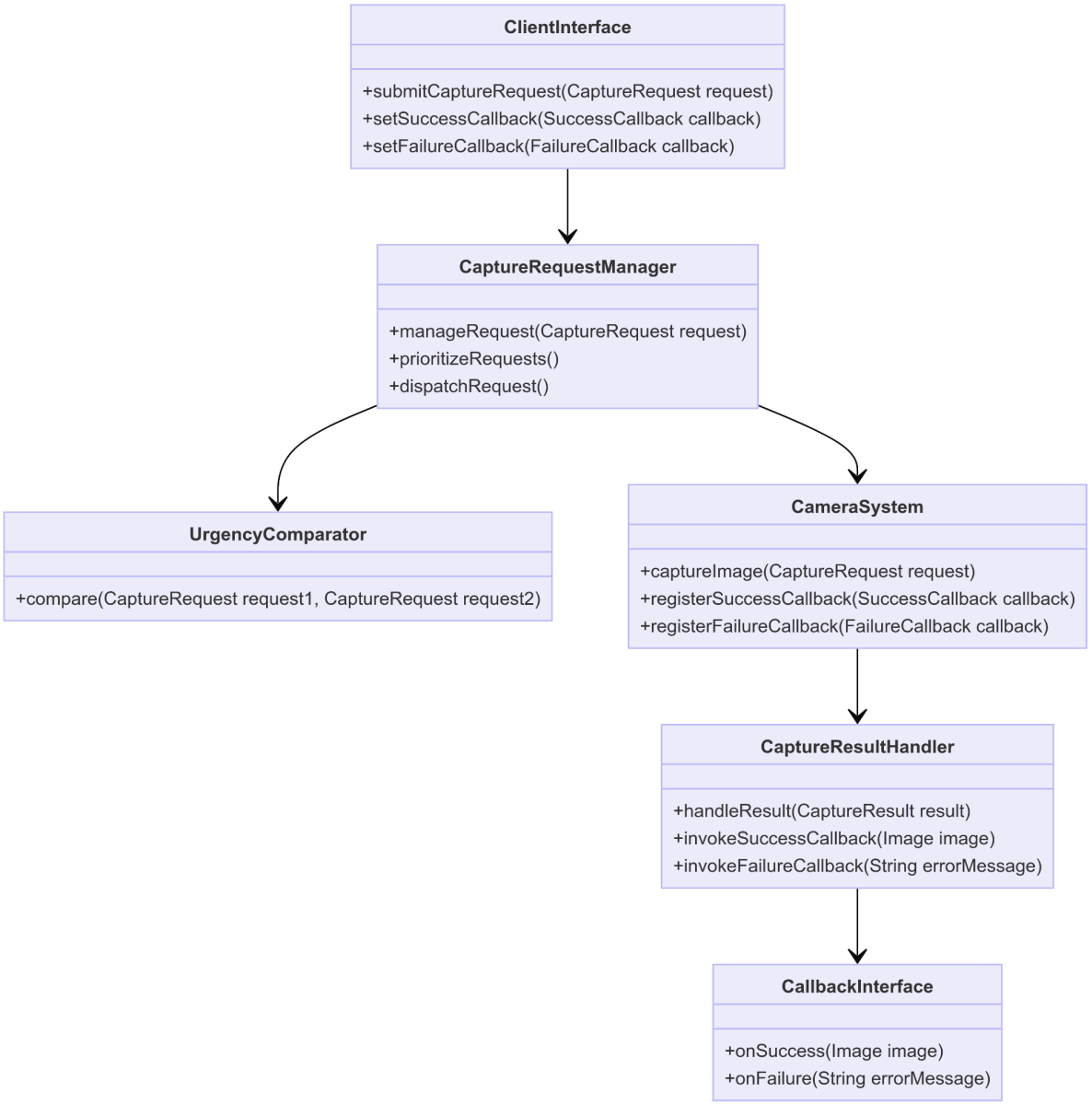
- handleResult(): Handles the result of the capture.
- invokeSuccessCallback(): Invokes the success callback with the captured image.
- invokeFailureCallback(): Invokes the failure callback with an error message.

Callback Interface:

- onSuccess(): Callback method invoked on successful image capture.
- onFailure(): Callback method invoked on image capture failure.



Low-Level Design of Camera System



Class Diagram

```
class ClientInterface {
    +submitCaptureRequest(CaptureRequest request)
    +setSuccessCallback(SuccessCallback callback)
    +setFailureCallback(FailureCallback callback)
}

class CaptureRequestManager {
    +manageRequest(CaptureRequest request)
    +prioritizeRequests()
```

```
    +dispatchRequest()
}

class UrgencyComparator {
    +compare(CaptureRequest request1, CaptureRequest request2)
}

class CameraSystem {
    +captureImage(CaptureRequest request)
    +registerSuccessCallback(SuccessCallback callback)
    +registerFailureCallback(FailureCallback callback)
}

class CaptureResultHandler {
    +handleResult(CaptureResult result)
    +invokeSuccessCallback(Image image)
    +invokeFailureCallback(String errorMessage)
}

class CallbackInterface {
    +onSuccess(Image image)
    +onFailure(String errorMessage)
}

ClientInterface --> CaptureRequestManager
CaptureRequestManager --> UrgencyComparator
CaptureRequestManager --> CameraSystem
CameraSystem --> CaptureResultHandler
CaptureResultHandler --> CallbackInterface
```

Sequence Diagram

```
participant ClientA participant ClientB participant CaptureRequestManager

participant UrgencyComparator participant CameraSystem participant

CaptureResultHandler participant SuccessCallback participant FailureCallback

ClientA->>CaptureRequestManager: submitCaptureRequest (High Urgency)

ClientB->>CaptureRequestManager: submitCaptureRequest (Low Urgency)

CaptureRequestManager->>UrgencyComparator: prioritize requests

UrgencyComparator->>CaptureRequestManager: return priority order
```

CaptureRequestManager->>CameraSystem: dispatch high priority request (ClientA)

CameraSystem->>CaptureResultHandler: return result (Success/Failure)

CaptureResultHandler->>SuccessCallback: invoke success callback (if successful)

CaptureResultHandler->>FailureCallback: invoke failure callback (if failed)

CaptureRequestManager->>CameraSystem: dispatch next request (ClientB)

CameraSystem->>CaptureResultHandler: return result (Success/Failure)

CaptureResultHandler->>SuccessCallback: invoke success callback (if successful)

CaptureResultHandler->>FailureCallback: invoke failure callback (if failed)

Design Patterns and SOLID Principles
Design Patterns:

1. **Observer Pattern**: Used for success and failure callbacks.
2. **Strategy Pattern**: Used for urgency comparison strategy.

SOLID Principles:

1. **Single Responsibility Principle (SRP)**: Each class has a single responsibility.
2. **Open/Closed Principle (OCP)**: System is open for extension (new urgency strategies) but closed for modification.
3. **Liskov Substitution Principle (LSP)**: Subtypes (e.g., different callback implementations) can replace base types.
4. **Interface Segregation Principle (ISP)**: Clients are not forced to depend on methods they do not use.
5. **Dependency Inversion Principle (DIP)**: High-level modules (CaptureRequestManager) depend on abstractions (CameraSystem) rather than low-level modules.

API Design

Client Interface

Endpoints

1. **Submit Capture Request**
    - **Endpoint**: POST /capture-request
    - **Description**: Submits a request to capture an image.

- **Request Body**:jsonCopy code

```
{
  "clientId": "string",
  "urgencyLevel": "int",
  "callbackUrl": "string"
}
```

- **Response**:
  - 200 OK on success
  - 400 Bad Request if the request is malformed
  - 500 Internal Server Error if an error occurs

2. **Set Success Callback**
   - **Endpoint**: POST /callbacks/success
   - **Description**: Registers a callback for successful image capture.
   - **Request Body**:jsonCopy code

```
{
  "clientId": "string",
  "callbackUrl": "string"
}
```

   - **Response**:
     - 200 OK on success
     - 400 Bad Request if the request is malformed
     - 500 Internal Server Error if an error occurs
3. **Set Failure Callback**
   - **Endpoint**: POST /callbacks/failure
   - **Description**: Registers a callback for failed image capture.
   - **Request Body**:jsonCopy code

```
{
  "clientId": "string",
  "callbackUrl": "string"
}
```

- **Response**:
    - 200 OK on success
    - 400 Bad Request if the request is malformed
    - 500 Internal Server Error if an error occurs

Capture Request Manager

Methods (Internal, not exposed via API)

1. manageRequest(request: CaptureRequest): void
    - **Description**: Manages incoming capture requests.
    - **Parameters**:
        - request (CaptureRequest): The request object containing details of the capture request.
2. prioritizeRequests(): void
    - **Description**: Prioritizes requests based on urgency.
    - **Parameters**: None
3. dispatchRequest(request: CaptureRequest): void
    - **Description**: Dispatches requests to the Camera System.
    - **Parameters**:
        - request (CaptureRequest): The request object containing details of the capture request.

Urgency Comparator

Methods (Internal, not exposed via API)

1. compare(request1: CaptureRequest, request2: CaptureRequest): int
    - **Description**: Compares the urgency of different requests.
    - **Parameters**:
        - request1 (CaptureRequest): The first request to compare.
        - request2 (CaptureRequest): The second request to compare.
    - **Returns**: An integer indicating the comparison result (negative if request1 is less urgent, zero if they are equally urgent, positive if request2 is more urgent)

Camera System

Endpoints (Typically, these would be internal methods, not directly exposed via REST API)

1. **Capture Image**
    - **Endpoint**: POST /camera/capture
    - **Description**: Captures an image asynchronously.
    - **Request Body**:jsonCopy code

      { "requestId": "string"}

      Response:

        - 202 Accepted on success
        - 400 Bad Request if the request is malformed
        - 500 Internal Server Error if an error occurs
        -
2. **Register Success Callback**

- **Endpoint**: POST /camera/success-callback
- **Description**: Registers a callback for successful capture.
- **Request Body**:jsonCopy code

```
{

 "requestId": "string",

 "callbackUrl": "string"

}
```

- **Response**:
  - 200 OK on success
  - 400 Bad Request if the request is malformed
  - 500 Internal Server Error if an error occurs

3. **Register Failure Callback**
   - **Endpoint**: POST /camera/failure-callback
   - **Description**: Registers a callback for capture failure.
   - **Request Body**:jsonCopy code

```
{

 "requestId": "string",

 "callbackUrl": "string"

}
```

   - **Response**:
     - 200 OK on success
     - 400 Bad Request if the request is malformed
     - 500 Internal Server Error if an error occurs

Capture Result Handler

Endpoints (Typically, these would be internal methods, not directly exposed via REST API)

1. **Handle Result**
   - **Endpoint**: POST /result
   - **Description**: Handles the result of the capture.
   - **Request Body**:jsonCopy code

```
{
```

```
  "requestId": "string",

  "status": "string", // success or failure

  "imageData": "base64String", // only if success

  "errorMessage": "string" // only if failure

}
```

- **Response**:
  - 200 OK on success
  - 400 Bad Request if the request is malformed
  - 500 Internal Server Error if an error occurs

2. **Invoke Success Callback**
   - **Endpoint**: POST /result/success
   - **Description**: Invokes the success callback with the captured image.
   - **Request Body**:jsonCopy code

```
{

  "callbackUrl": "string",

  "imageData": "base64String"

}
```

   - **Response**:
     - 200 OK on success
     - 400 Bad Request if the request is malformed
     - 500 Internal Server Error if an error occurs
3. **Invoke Failure Callback**
   - **Endpoint**: POST /result/failure
   - **Description**: Invokes the failure callback with an error message.
   - **Request Body**:jsonCopy code

```
{

  "callbackUrl": "string",

  "errorMessage": "string"

}
```

   - **Response**:

- 200 OK on success
- 400 Bad Request if the request is malformed
- 500 Internal Server Error if an error occurs

Callback Interface

Methods

1. onSuccess(imageData: base64String): void
    - **Description**: Callback method invoked on successful image capture.
    - **Parameters**:
        - imageData (base64String): The captured image in base64 format.
2. onFailure(errorMessage: string): void
    - **Description**: Callback method invoked on image capture failure.
    - **Parameters**:
        - errorMessage (string): The error message indicating the cause of failure.

DATABASE

```
Table to store clients
CREATE TABLE Clients (
    clientId VARCHAR(255) PRIMARY KEY,
    clientName VARCHAR(255) NOT NULL
);

-- Table to store capture requests
CREATE TABLE CaptureRequests (
    requestId VARCHAR(255) PRIMARY KEY,
    clientId VARCHAR(255),
    urgencyLevel INT NOT NULL,
    status VARCHAR(50) CHECK (status IN ('pending', 'in_progress', 'completed', 'failed')),
    requestTime TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (clientId) REFERENCES Clients(clientId)
);

-- Table to store callbacks
CREATE TABLE Callbacks (
    callbackId SERIAL PRIMARY KEY,
    clientId VARCHAR(255),
    callbackType VARCHAR(50) CHECK (callbackType IN ('success', 'failure')),
    callbackUrl VARCHAR(255) NOT NULL,
    FOREIGN KEY (clientId) REFERENCES Clients(clientId)
);

-- Table to store capture results
CREATE TABLE CaptureResults (
    resultId SERIAL PRIMARY KEY,
    requestId VARCHAR(255),
    status VARCHAR(50) CHECK (status IN ('success', 'failure')),
    imageData TEXT,
```

```
errorMessage TEXT,
resultTime TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (requestId) REFER
```