# Requirement Document

## Project Title: Pixel Stream Resource Manager

### Introduction

The Pixel Stream Resource Manager is a FastAPI application designed to manage pixel streaming resources. It allows clients to request and release streams, and it maintains a record of active streams and the clients using them.

### Functional Requirements

1. Stream Management

- Request Stream:

- Endpoint: `/getstream`

- Method: `POST`

- Parameters:

- `requester\_id` (str): The ID of the requester.

- `service` (str): The streaming service ("AWS" or "StreamPixel").

- Description: Allocates an available stream to the requester. If the requester already has an active stream, it returns the existing stream's IP. Otherwise, it allocates a new stream and marks it as active.

- Response:

- Success: Returns the IP of the assigned stream.

- Failure: Returns a 404 error if no available streams are found or a 400 error if the requester already has an active stream with another service.

- Check Available Stream:

- Endpoint: `/check\_stream`

- Method: `GET`

- Parameters:

- `service` (str): The streaming service ("AWS" or "StreamPixel").

- Description: Checks if there are any available streams for the specified service.

- Response:

- Success: Returns the stream ID and IP of an available stream.

- Failure: Returns a message indicating no available streams found.

- Release Stream:

- Endpoint: `/release\_stream`

- Method: `POST`

- Parameters:

- `requester\_id` (str): The ID of the requester.

- Description: Releases the stream associated with the requester, marking it as available again.

- Response:

- Success: Returns a message indicating the stream has been released and is available again.

- Failure: Returns a 404 error if the requester does not have any active streams.

2. Stream Monitoring

- List Active Streams:

- Endpoint: `/active\_streams`

- Method: `GET`

- Description: Returns a list of all active streams, categorized by service.

- Response: A dictionary where keys are services and values are dictionaries mapping stream IDs to IP addresses.

- List Requester Streams:

- Endpoint: `/requester\_streams`

- Method: `GET`

- Description: Returns a list of all requester IDs and the streams assigned to them, including stream IDs, services, and IP addresses.

- Response: A dictionary where each requester ID is mapped to its stream ID, service, and IP address.

Non-Functional Requirements

1. Performance: The application should handle requests efficiently and return responses within acceptable time limits.

2. Scalability: The system should be able to manage a large number of streams and requesters without significant degradation in performance.

3. Security: Proper error handling should be in place to prevent information leakage. Sensitive data such as API keys should be managed securely.

4. Reliability: The system should maintain consistency in managing stream states (active and available).

Environment Setup

1. Python Libraries: The application requires several Python libraries which can be installed via `pip`.

- `fastapi`

- `uvicorn`

- `python-dotenv`

- `boto3`

- `orjson`

2. AWS Configuration: If AWS is used for streaming, appropriate AWS credentials should be configured and available in the environment variables.

3. Docker: The application should be containerized using Docker for consistent deployment across different environments.