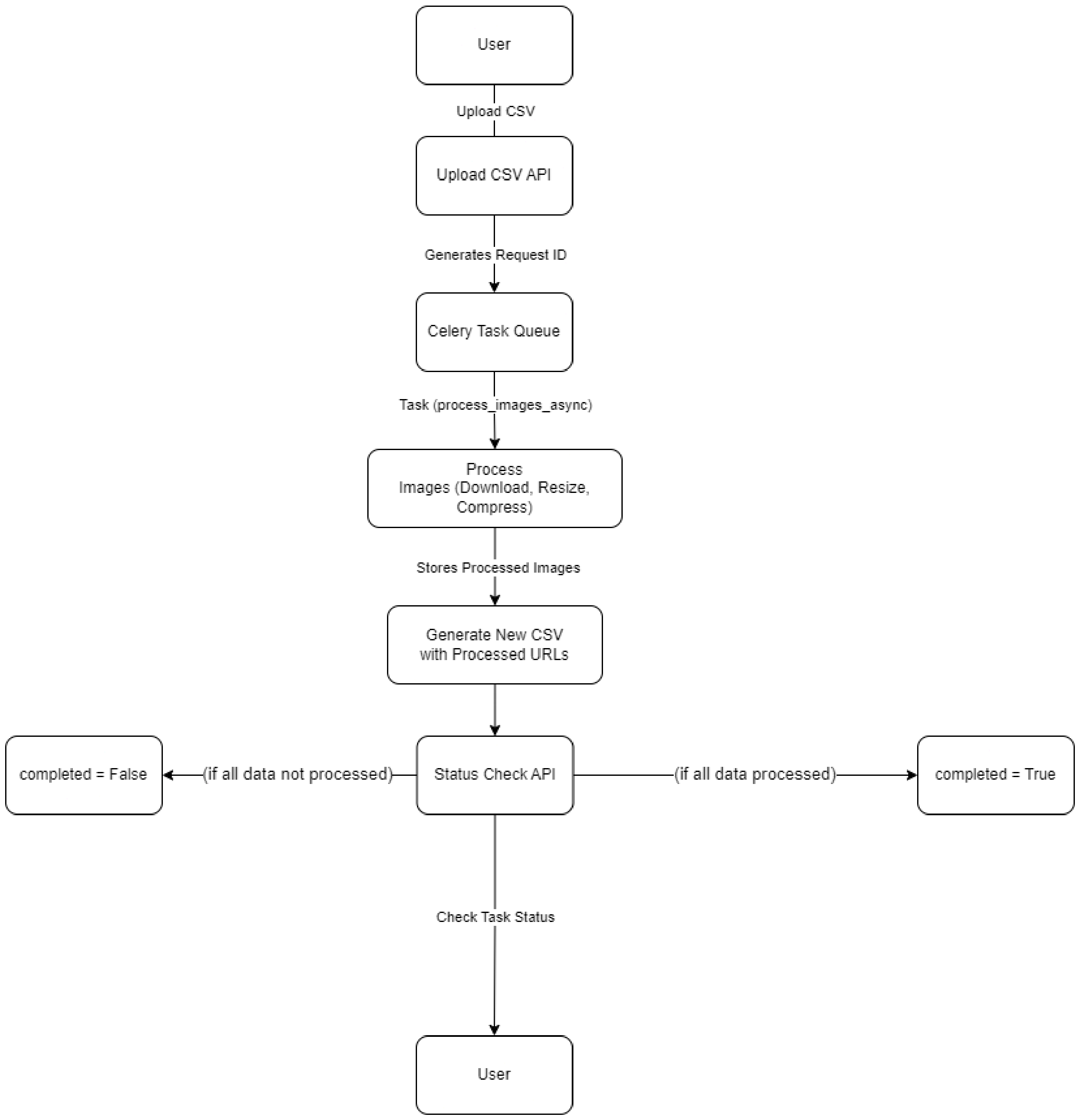
Low-Level Design (LLD) Document

1. Introduction

This document provides a detailed low-level design for the image processing system. The project focuses on efficiently handling and processing image data, allowing users to upload images, process them asynchronously, and retrieve processed results. The system integrates various components to manage image resizing, compression, and metadata storage.

The design outlines the system architecture, component functions, and interactions. The accompanying diagram visually represents the components and their relationships, providing a comprehensive view of the system's structure and data flow.

2. System Architecture Diagram



3. Component Descriptions

3.1 User Interface

- **Role**: Provides the front-end for user interaction.
- **Function**:
  - Displays data to users.
  - Collects user input.
  - Communicates with the application server via API calls.

        - Technologies: Python-Django.

3.2 Application Server

- **Role**: Hosts the application logic and handles business processes.
- **Function**:
    - Processes user requests.
    - Executes business logic.
    - Interacts with the database and external services.
    - Manages API requests and responses.
    - Technologies: Django 5.0, Python, REST APIs.

3.3 Database

- **Role**: Stores and manages application data.
- **Function**:
    - Performs CRUD operations on data.
    - Handles schema, relationships, and indexing for efficient data retrieval.
    - Technologies: SQLite

3.4 Image Processing Service

- **Role**: Handles the processing and manipulation of images.
- **Function**:
    - Receives images for processing.
    - Applies transformations such as resizing and compression.
    - Stores or returns processed images.
    - Technologies: PIL/Pillow, Bytesio, Celery, Custom Processing Logic.

3.5 Logging Service

- **Role**: Monitors and logs system activities.
- **Function**:
    - Captures logs for debugging, auditing, and performance monitoring.
    - Provides insights into system behavior and errors.
    - Technologies: Python Logging.

3.6 External APIs/Services

- **Role**: Interfaces with third-party services.
- **Function**:
    - Sends and receives data to/from external systems such as image storage services.
    - Technologies: RESTful APIs, HTTP/HTTPS.

4. Data Flow Diagrams

(Describe data flow here or add relevant diagrams)

4.1 Data Flow

- **User Interface**: Sends requests to the Application Server.
- **Application Server**:
    - Receives and processes requests.
    - Interacts with the Database to retrieve or store data.
    - Communicates with the Image Processing Service for image-related tasks.
- **Image Processing Service**: Processes images as requested by the Application Server.
- **Database**: Stores and retrieves data as needed by the Application Server.
- **External APIs/Services**: Interacts with external systems to exchange data.

5. API Specifications

5.1 Endpoint: `/api/upload/`

- **Method**: POST
- **Description**: Uploads an image for processing.
- **Parameters**:
    - `file` (File): The CSV file to be processed.
- **Response**:
    - `status` (String): Success or error status code/message.
    - `request_id` (String): Request id of the processed CSV file.

5.2 Endpoint: `/api/status/?request_id=<request_id>`

- **Method**: GET
- **Description**: Gets status of the entered request ID.
- **Parameters**:
    - `request_id` (String): Request ID of the file to be processed.
- **Response**:
    - `status` (String): Success or error code/message.
    - `request_id` (String): request ID of the processed file.
    - `Created_at` (String): created time of the process.
    - `Completed` (Bool): Status(True/False) of file processing

6. Database Schema

6.1 Tables

- **Product**
  - `id` (Integer, Primary Key)
  - `serial_number` (Integer)
  - `name` (String)
  - `input_image_url` (String)
  - `output_image_url` (String)
  - `upload_time` (Datetime)
  - `Processed` (Bool)

- **Processing Request**
  - `id` (Integer, Primary Key)
  - `name` (String)
  - `result_file` (File)
  - `Completed` (Bool)
  - `Created_at` (Datetime)

7. Image Processing Specifications

- **Resizing**: Reduce image dimensions by 50%.
- **Compression**: Reduce image quality by 50%.
- **Formats**: JPEG, PNG.

  #FORMATS AND RESIZING IS FLEXIBLE

8. Logging and Monitoring

- **Logging**:
  - System logs for debugging and performance monitoring.
  - Error logs for capturing exceptions and issues.
- **Monitoring**:
  - Tracks system health and performance metrics.
  - Uses Python logs for logging and monitoring.