Table of Content

1. Introduction
   1. Overview

* Use-Case Diagram
  1. Technologies Used

1. Setup and Installation
   1. Prerequisites

* Version of Express.js, node.js & MongoDB
* Modules/Middleware Used
  1. Installation Guide
  2. Running the Application

1. Architecture and Design
   1. System Architecture

* Structure Diagram
  1. Database Schema
* ER Diagram
* Data Dictionary
  1. System Overview
* State Diagram
* Assumption and Limitation (1 album for each user. Max 2 raw photos)
* Brief Description of each Process

1. User Management
   1. Sign Up
   2. Login and Logout
   3. User Profile Edit
   4. Un-subscription
2. Photo Management
   1. Photo Upload
   2. Photo Retrieval and Display
3. APIs and Routes
   1. RESTful API Endpoints
   2. Route Handlers
4. Security Control
   1. Password Hashing
   2. Protection of Photo from Direct Access
5. Future Enhancements (???)

# Copilot Suggestion for Section 6

**Overview:**

* Description of what RESTful APIs are and their importance in a web application.
* Explanation of how your photo-sharing application uses RESTful APIs to handle HTTP requests and responses.

**Common HTTP Methods:**

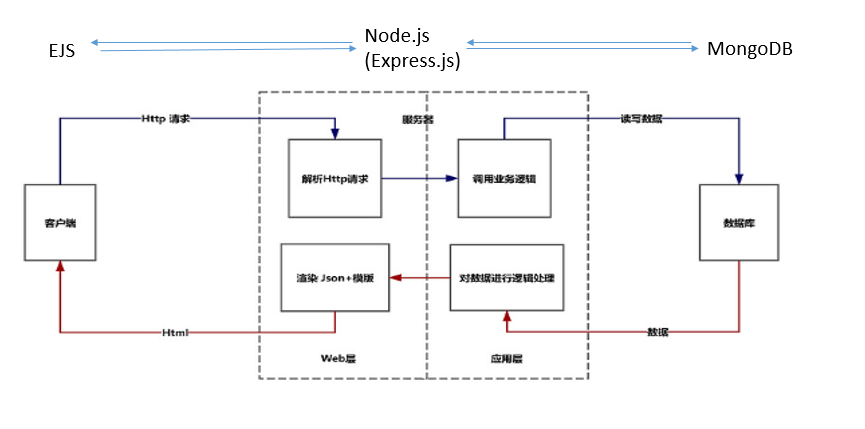
* **GET:** Retrieve data from the server.
* **POST:** Send new data to the server.
* **PUT:** Update existing data on the server.
* **DELETE:** Remove data from the server.

**API Endpoint Structure:**

* **Base URL:** The root URL for all API endpoints, e.g., https://yourapp.com/api/v1.
* **Endpoint Path:** The specific resource paths, e.g., /photos, /users, /comments.

**Example Endpoints:**

1. **User Authentication:**
   * **POST /api/v1/register:** Register a new user.
   * **POST /api/v1/login:** Authenticate a user and return a token.
2. **Photo Management:**
   * **GET /api/v1/photos:** Retrieve a list of all photos.
   * **POST /api/v1/photos:** Upload a new photo.
   * **GET /api/v1/photos/:id:** Retrieve a single photo by ID.
   * **PUT /api/v1/photos/:id:** Update details of a specific photo.
   * **DELETE /api/v1/photos/:id:** Delete a specific photo.



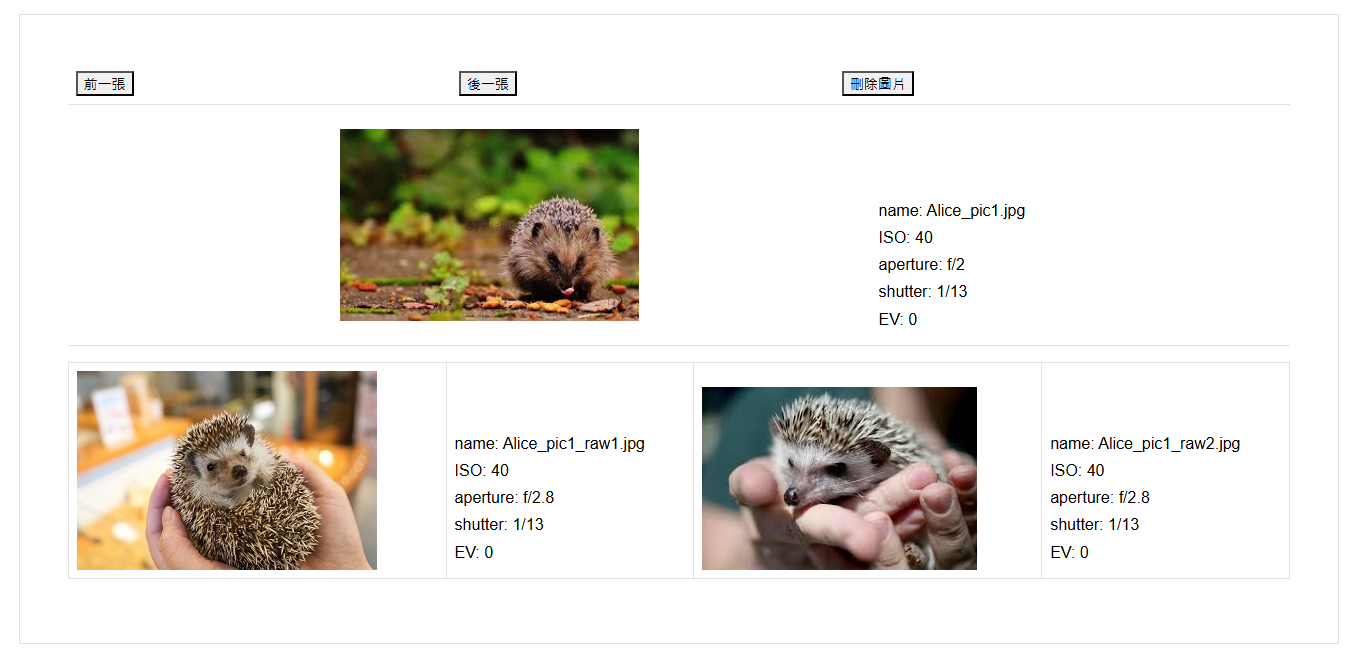
This system architecture for a web application using EJS (Embedded JavaScript), Node.js, Express.js, and MongoDB.

Client (Front End), These are used HTML/CSS/JavaScript to build the structure, style, and behavior of the web pages. Used Embedded JavaScript (EJS) templates for server-side rendering of HTML. Used Fetch API to make HTTP requests to the server for data.

Server, Used Node.js a lightweight framework to build web applications and APIs. Used Router Layer Defines the application's endpoint routes and handles HTTP requests. Used Controller Layer Contains the business logic and interacts with the Model Layer to process requests. Used Model Layer defines the data schema and interacts with MongoDB to perform CRUD operations.

Database, Use a NoSQL database MongoDB to store application datas . There are include two collections that stores user information and stores information related to photos.

Photo detail



This is a photo details page.

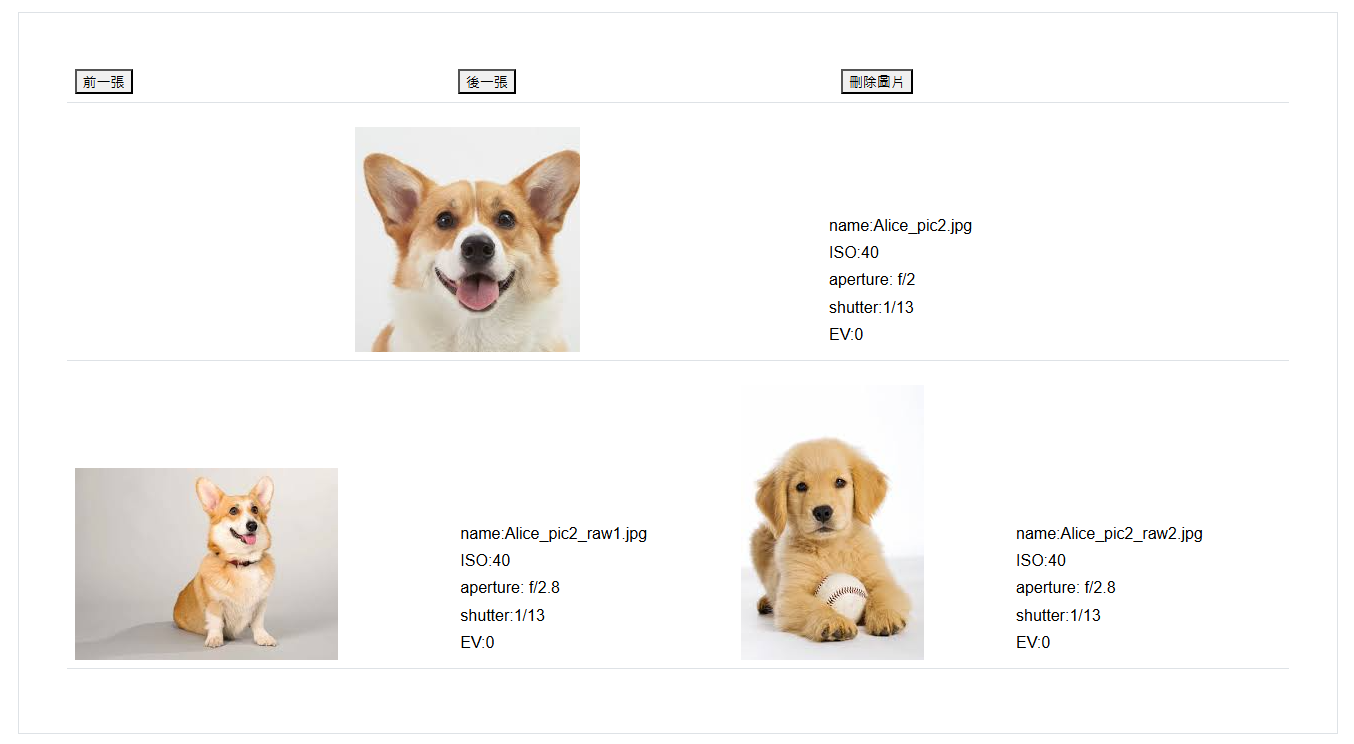
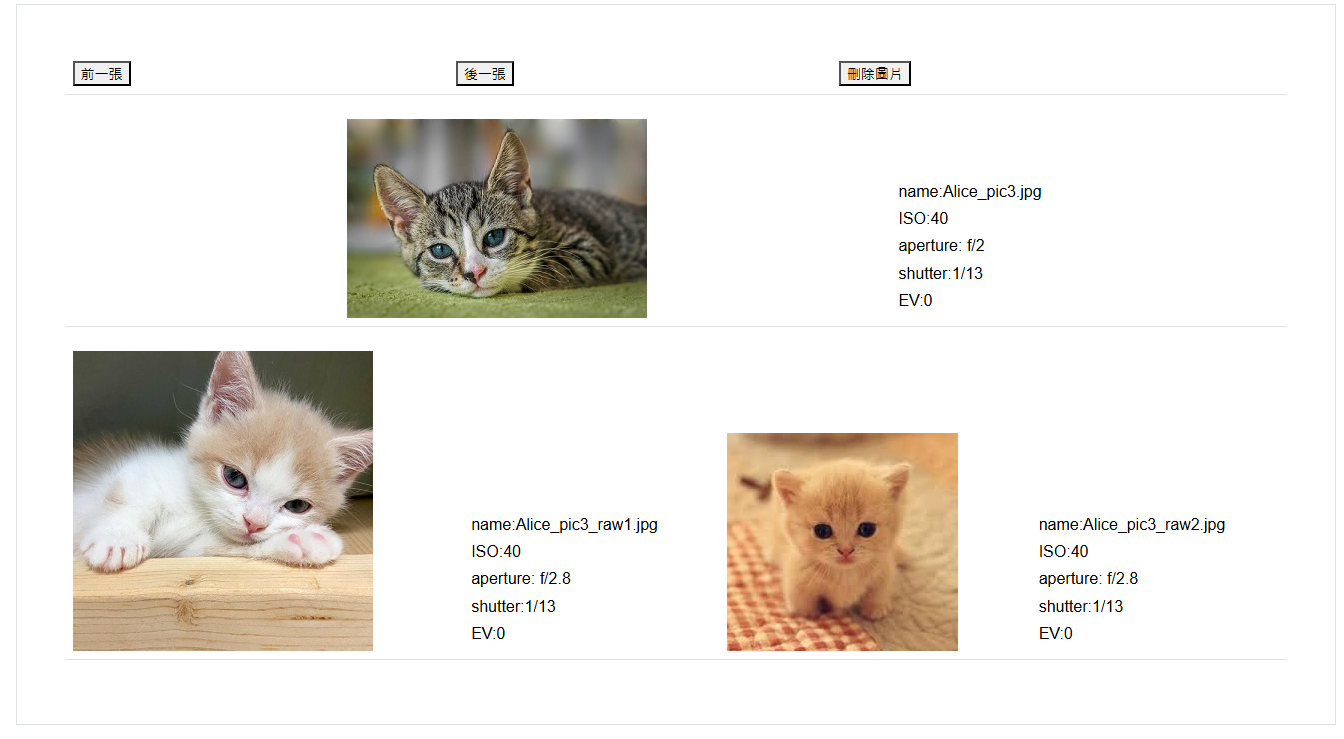
This page included three buttons(" previous picture"," next picture"," delete picture") , one main picture ,two raw pictures and some parameters for pictures.

Three buttons are displayed at the top of the page and the main picture and some parameters of the main picture, and the two raw pictures and some parameters of the raw picture below the main picture.

Click " previous picture", It will display the previous picture. If this picture is the first picture, "Already the first picture" will pop up at the page top.

Click " next picture", It will display the next picture. If this picture is the last picture, "Already the last picture" will pop up at the page top.

Click "Delete Picture" , It will delete this picture and the two raw pictures and parameters under this picture. Only pictures uploaded by yourself can see this button . In this way, users who log in to the system can delete their own pictures, but not other people's pictures.



Get list/photodetail: Retrieve a single photo details with its raw photos from photo list by ID.

Get list/delephoto: Retrieve a single photo details with its raw photos from photo list by ID.

/photodetail