Table of Content

1. Introduction (Amanda)
   1. Overview

* Use-Case Diagram
  1. Technologies Used

1. Setup and Installation
   1. Prerequisites (Amanda)

* Version of Express.js, node.js & MongoDB
* Modules Used
  1. Installation Guide (Wah)
  2. Running the Application (Wah)

1. Architecture and Design
   1. System Architecture (Moon)

* Structure Diagram
  1. Database Schema (Vincent - Done)
* ER Diagram
* Data Dictionary
  1. System Overview (Vincent)
* State Diagram
* Assumption and Limitation
* Brief Description of System Functionality

1. User Management (Vincent)
   1. Sign Up
   2. Login and Logout
   3. User Profile Edit
   4. Un-subscription
2. Photo Management
   1. Photo Upload (Wah)
   2. Photo Retrieval and Display
      1. Album List (Megan)
      2. Photo List (Amanda)
      3. Photo Details (Moon)
3. APIs and Routes (All)
   1. RESTful API Endpoints
   2. Route Handlers
4. Security Control (Vincent - Done)
   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.

# Data Dictionary

Collection: user\_profile

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Description |
| login\_id | String | Unique identity used to enter into the application |
| display\_name | String | Name of the user used to show in application |
| password | String | Hash value of secret key used by the user to log-in to system |
| profile\_photo | String | Filename of an image file used by the user as his/her profile picture |

Collection: photo\_collection

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Description |
| login\_id | String | Used as identity of album folder owned by the user |
| filename | String | Filename of a photo uploaded by the user |
| ISO | String | Technical aspects of a photo |
| aperture | String |
| shutter | String |
| EV | String |
| likeCount | Integer | Total number of user response like a photo |
| likeBy | Array of string | Login ID of user(s) who response like a photo |
| raw1 | Raw photo object | raw photo used to compose the photo in this record |
| raw2 | Raw photo object |

Object: raw photo

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Description |
| filename | String | Filename of a raw photo uploaded by the user |
| ISO | String | Technical aspects of a photo |
| aperture | String |
| shutter | String |
| EV | String |

# Assumption & Limitation

As a pilot project, few assumption has been made to simplify implementation. Those assumption is listed as below: -

* Each user is assumed to have one and only one album
* Each multi-exposure photo has maximum of 2 raw photos
* This project is assumed as low usage volume application

Due to above assumption, system inherits with below limitation: -

* Each photo belongs to one and only one album. It cannot be shared among different album.
* System will create an album for each new user, and cannot be changed. User do not have rights to create any additional album.
* System only provides 2 slots for each photo to upload raw photos in photo upload module
* No performance issue is considered when design and implement this application

# Security Control

To prevent application from unauthorized or unauthenticated use, few security measurements are enforced.

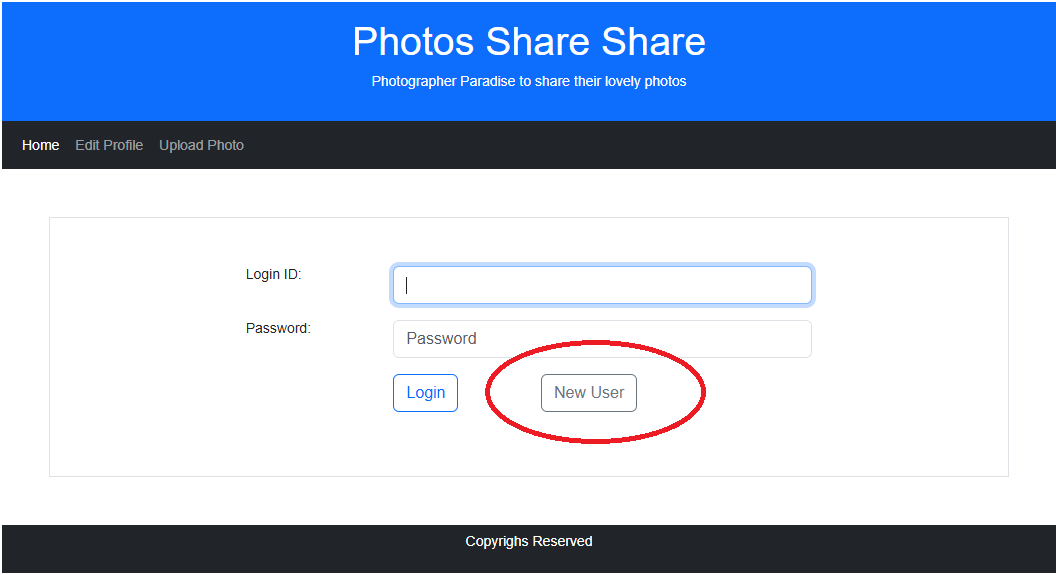
* User password is hashed before stored into database. It prevents anyone to spy on someone password by directing retrieving data from database.
* As /public folder is opened for anyone to access. If image files are kept under this folder, anyone can by-pass the application to access those files directly. To prevent this, A /data folder separated from /public folder is used for image files storage. Outsiders has no access rights to this folder. When application accesses those files, API is used to re-route access path to /data folder.

# Brief Description of System Functionality

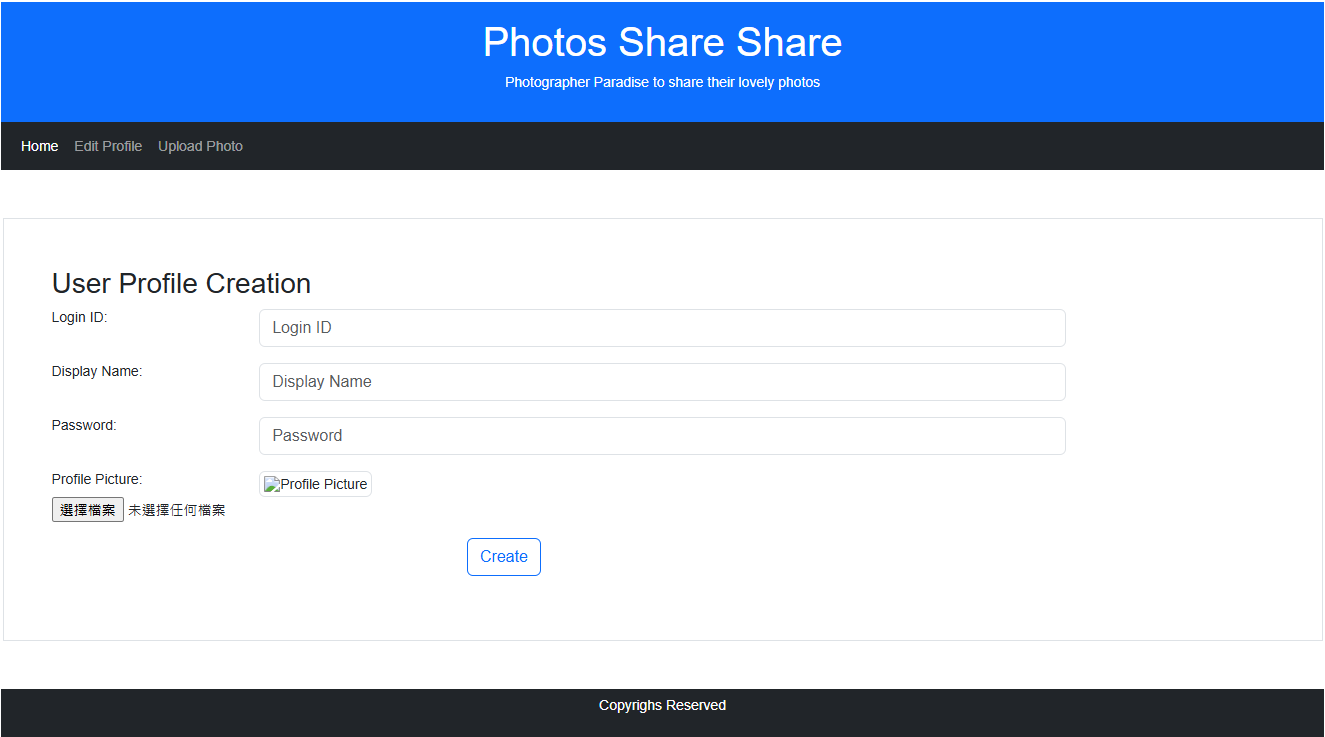
* User Account Management
  + System allows user to create/modify/remove his own user record. Except for user ID, system will also record his chosen name showed in system and profile picture (if provided).
* Album Management
  + Each user has its own album. He can upload his personal photo into his own album. Photo in the album can also be removed.
* Photo Browsing
  + User can browse all albums recorded in the system. In photo detail level, user can also view raw photos of those multi-exposure photo. He can click like icon to show his appreciation on any photo.

# User Management

### Sign Up



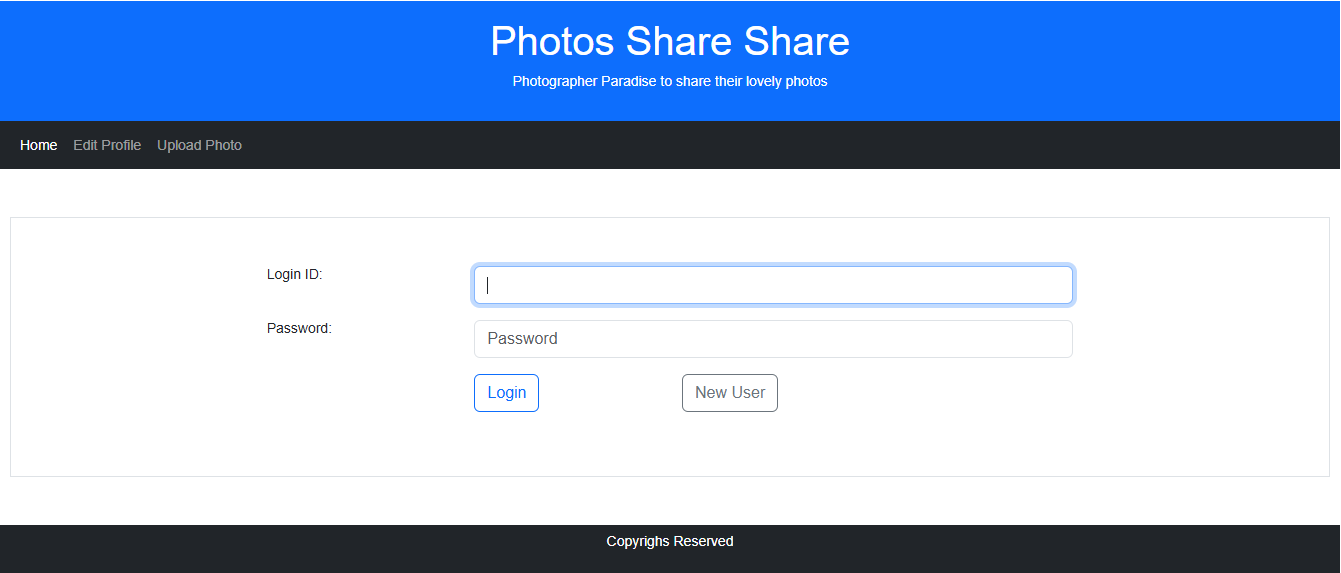
For new user who do not have account in the system, click <New User> button in log-in screen. It will lead to below new user creation screen.



Except for profile picture, all fields are mandatory. After user input all required information, press <Create> button to generate a user record for him. If success, system will direct to login screen then.

### Login and Logout

To log-in into system, enter log-in and password in log-in screen. Then press <Log-in> to proceed



In main menu, there is a <Logout> menu item. Press it to log-out from system. System will then direct to log-in screen.



### User Profile Edit

Created user profile can be edited by its owner. After log-in to system, press <Edit Profile> menu option will lead to user profile edit screen, similar as below: -