DB DESIGN

1. User Management Table (for authentication and profile)

Table Name: users

Column Name	**Data Type**	**Description**
id	INT (PK)	Unique identifier for each user. Primary Key.
email	VARCHAR(255)	User's email for login, used for communication and identification.
phone	VARCHAR(15)	Optional phone number for contact.
password_hash	VARCHAR(255)	Encrypted password for user authentication.
role	ENUM('CUSTOMER', 'PHOTOGRAPHER', 'EVENT_ORGANIZER')	Role-based access control, differentiates user types.
profile_pic	VARCHAR(255)	URL to the user's profile picture (if applicable).
created_at	TIMESTAMP	Timestamp for account creation.
updated_at	TIMESTAMP	Timestamp for last account update.

Explanation:

- **id** is the primary key, uniquely identifying each user.
- **email** and **phone** are for user authentication. You can implement multi-factor authentication based on these fields.
- **password_hash** stores the encrypted version of the user's password.
- **role** helps differentiate between customers, photographers, and event organizers.
- **profile_pic** stores the URL for the user's profile image.
- **created_at** and **updated_at** track the account's creation and modification times.

---

2. Event Management Table

Table Name: events

Column Name	**Data Type**	**Description**
id	INT (PK)	Unique event identifier.

Column Name	Data Type	Description
name	VARCHAR(255)	Name of the event (e.g., "Wedding", "Corporate Shoot").
event_type	ENUM('WEDDING', 'BIRTHDAY', 'CORPORATE', 'FASHION', 'OTHER')	Type of event for categorization.
location	VARCHAR(255)	Event location (for event-specific search).
start_date	TIMESTAMP	Date and time when the event starts.
end_date	TIMESTAMP	Date and time when the event ends.
status	ENUM('PENDING', 'COMPLETED', 'CANCELLED')	Event status (used for sorting/filtering events).
organizer_id	INT (FK)	Foreign key referring to users.id for event organizer.
created_at	TIMESTAMP	Timestamp when the event was created.
updated_at	TIMESTAMP	Timestamp when the event was last updated.

Explanation:

- **id**: Primary key for each event.
- **event_type** categorizes events for filtering (Wedding, Birthday, etc.).
- **location** stores where the event is happening (e.g., city name, venue).
- **start_date** and **end_date** are necessary for managing event schedules.
- **status** allows users and admins to track the current state of the event (Pending, Completed, etc.).
- **organizer_id** links the event to the **Event Organizer** (foreign key).

---

3. Media Management Table

Table Name: media

Column Name	Data Type	Description
id	INT (PK)	Unique identifier for each media file (photo/video).
event_id	INT (FK)	Foreign key referring to the events.id table, linking media to an event.
file_url	VARCHAR(255)	URL to the media file stored in cloud (e.g., S3).
media_type	ENUM('IMAGE', 'VIDEO')	Type of media (image or video).
file_size	INT	Size of the media file in bytes.
created_at	TIMESTAMP	Date and time when the media was uploaded.
updated_at	TIMESTAMP	Date and time when the media metadata was last updated.
status	ENUM('PENDING', 'PROCESSED', 'FAILED')	Status of the media file (e.g., processing, complete).

Explanation:

- **id**: Unique identifier for each media.
- **event_id**: Links media to a specific event.
- **file_url**: Stores the S3/Cloud URL where the media is stored.
- **media_type**: Differentiates between photos and videos.
- **file_size**: Used for file management, tracking size limits or quotas.
- **status**: Tracks the processing state (e.g., uploaded, processed, failed).

---

4. Booking & Request Management Table

Table Name: bookings

Column Name	Data Type	Description
id	INT (PK)	Unique booking identifier.
event_id	INT (FK)	Foreign key referring to events.id, linking booking to an event.
customer_id	INT (FK)	Foreign key referring to users.id, linking booking to a customer.

Column Name	Data Type	Description
photographer_id	INT (FK)	Foreign key referring to users.id, linking booking to a photographer.
status	ENUM('PENDING', 'CONFIRMED', 'COMPLETED', 'CANCELLED')	Status of the booking.
booking_date	TIMESTAMP	Date and time when the booking was made.
payment_status	ENUM('PENDING', 'PAID', 'FAILED')	Payment status for the booking.
created_at	TIMESTAMP	Timestamp when the booking was created.

Explanation:

- **event_id**: Links the booking to a specific event.
- **customer_id** and **photographer_id**: Associate a customer with a photographer.
- **status**: Tracks the progress of the booking (Pending, Confirmed, etc.).
- **payment_status**: Allows for tracking if the payment has been processed.

---

5. Messaging System Table

Table Name: messages

Column Name	Data Type	Description
id	INT (PK)	Unique message identifier.
sender_id	INT (FK)	Foreign key referring to users.id, the sender of the message.
receiver_id	INT (FK)	Foreign key referring to users.id, the receiver of the message.
message	TEXT	The content of the message.
timestamp	TIMESTAMP	Time when the message was sent.
status	ENUM('SENT', 'READ', 'ARCHIVED')	Status of the message (sent, read, archived).

Explanation:

- **sender_id** and **receiver_id**: Represents the communication between users (customers, photographers, event organizers).

- **message**: The content of the message.
- **status**: Tracks whether the message has been read, archived, or just sent.

---

6. Photographer Profile Table

Table Name: photographer_profiles

Column Name	Data Type	Description
id	INT (PK)	Unique photographer profile identifier.
user_id	INT (FK)	Foreign key referring to users.id, linking profile to a user.
studio_name	VARCHAR(255)	Name of the studio or photography business.
specialty	VARCHAR(255)	Type of photography services offered (e.g., wedding, portrait).
pricing	TEXT	Details of the pricing for photography services.
portfolio_url	VARCHAR(255)	URL to the photographer's portfolio (hosted elsewhere).
created_at	TIMESTAMP	Timestamp when the profile was created.

Explanation:

- **user_id**: Links the photographer profile to a user in the users table.
- **studio_name**: Identifies the photographer's business/studio.
- **specialty** and **pricing**: Describe the type of services and their costs.
- **portfolio_url**: A link to the photographer's portfolio (if hosted externally).

---

7. Review & Rating Table

Table Name: reviews

Column Name	Data Type	Description
id	INT (PK)	Unique review identifier.
reviewer_id	INT (FK)	Foreign key referring to users.id, the user giving the review.
reviewee_id	INT (FK)	Foreign key referring to users.id, the user receiving the review.
rating	INT	Rating given (1 to 5 stars).
comment	TEXT	Review comment from the reviewer.
created_at	TIMESTAMP	Timestamp when the review was posted.

Explanation:

- **reviewer_id** and **reviewee_id**: Allows users to leave reviews for photographers or event organizers.
- **rating**: A numeric rating (e.g., 1-5 stars).
- **comment**: The content of the review.

1. User Management Table (users)

```
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,        -- Unique identifier for each user
  email VARCHAR(255) NOT NULL,              -- User's email for login (unique)
  phone VARCHAR(15),                        -- Optional phone number for contact
  password_hash VARCHAR(255) NOT NULL,        -- Encrypted password for authentication
  role ENUM('CUSTOMER', 'PHOTOGRAPHER', 'EVENT_ORGANIZER') NOT NULL,  -- User's role
  profile_pic VARCHAR(255),                 -- URL to the user's profile picture
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- Account creation timestamp
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP -
- Timestamp for updates
);

-- Index for fast lookup of users by email
CREATE UNIQUE INDEX idx_email ON users(email);
```

---

2. Event Management Table (events)

```
CREATE TABLE events (
  id INT AUTO_INCREMENT PRIMARY KEY,        -- Unique event identifier
  name VARCHAR(255) NOT NULL,               -- Event name (e.g., Wedding, Corporate)
  event_type ENUM('WEDDING', 'BIRTHDAY', 'CORPORATE', 'FASHION', 'OTHER') NOT NULL,  --
Type of event
  location VARCHAR(255),                    -- Event location
  start_date TIMESTAMP NOT NULL,            -- Start date and time of the event
  end_date TIMESTAMP NOT NULL,              -- End date and time of the event
  status ENUM('PENDING', 'COMPLETED', 'CANCELLED') DEFAULT 'PENDING', -- Event status
  organizer_id INT NOT NULL,              -- Foreign key referring to users.id (Event Organizer)
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- Event creation timestamp
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
-- Event update timestamp
  FOREIGN KEY (organizer_id) REFERENCES users(id) -- Foreign key for event organizer
);

-- Index for searching events by organizer or status
CREATE INDEX idx_event_status ON events(status);
```

---

3. Media Management Table (media)

```
CREATE TABLE media (
  id INT AUTO_INCREMENT PRIMARY KEY,          -- Unique media identifier (photo/video)
  event_id INT NOT NULL,                  -- Foreign key to events.id (Media associated with an
event)
  file_url VARCHAR(255) NOT NULL,           -- URL of the media file (stored on cloud, e.g., S3)
  media_type ENUM('IMAGE', 'VIDEO') NOT NULL,   -- Type of media (photo or video)
  file_size INT NOT NULL,                 -- Size of the media file in bytes
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- Timestamp for when the media
was uploaded
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
-- Timestamp for media updates
  status ENUM('PENDING', 'PROCESSED', 'FAILED') DEFAULT 'PENDING', -- Processing status of
media
  FOREIGN KEY (event_id) REFERENCES events(id)  -- Foreign key to events table
);

-- Index for media retrieval by event
CREATE INDEX idx_media_event ON media(event_id);
```

---

4. Booking & Request Management Table (bookings)

```
CREATE TABLE bookings (
  id INT AUTO_INCREMENT PRIMARY KEY,        -- Unique booking identifier
  event_id INT NOT NULL,                -- Foreign key referring to events.id (Linking booking to an event)
  customer_id INT NOT NULL,             -- Foreign key referring to users.id (Linking booking to a customer)
  photographer_id INT NOT NULL,          -- Foreign key referring to users.id (Linking booking to a photographer)
  status ENUM('PENDING', 'CONFIRMED', 'COMPLETED', 'CANCELLED') DEFAULT 'PENDING',  -- Booking status
  booking_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- Timestamp when the booking was created
  payment_status ENUM('PENDING', 'PAID', 'FAILED') DEFAULT 'PENDING', -- Payment status
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- Booking creation timestamp
  FOREIGN KEY (event_id) REFERENCES events(id), -- Foreign key to event table
  FOREIGN KEY (customer_id) REFERENCES users(id), -- Foreign key to users table (customer)
  FOREIGN KEY (photographer_id) REFERENCES users(id) -- Foreign key to users table (photographer)
);


-- Index for booking search by customer or photographer
CREATE INDEX idx_booking_customer ON bookings(customer_id);
CREATE INDEX idx_booking_photographer ON bookings(photographer_id);
```

---

5. Messaging System Table (messages)

```
CREATE TABLE messages (
  id INT AUTO_INCREMENT PRIMARY KEY,        -- Unique message identifier
  sender_id INT NOT NULL,               -- Foreign key referring to users.id (Sender of the message)
  receiver_id INT NOT NULL,             -- Foreign key referring to users.id (Receiver of the message)
  message TEXT NOT NULL,                -- Content of the message
  timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- Timestamp when the message was sent
  status ENUM('SENT', 'READ', 'ARCHIVED') DEFAULT 'SENT', -- Message status (read, sent, archived)
  FOREIGN KEY (sender_id) REFERENCES users(id), -- Foreign key for sender
  FOREIGN KEY (receiver_id) REFERENCES users(id) -- Foreign key for receiver
);

-- Index for fast search by sender and receiver
CREATE INDEX idx_message_sender ON messages(sender_id);
CREATE INDEX idx_message_receiver ON messages(receiver_id);
```

---

6. Photographer Profile Table (photographer_profiles)

```
CREATE TABLE photographer_profiles (
  id INT AUTO_INCREMENT PRIMARY KEY,       -- Unique photographer profile identifier
  user_id INT NOT NULL,                -- Foreign key referring to users.id (Linking profile to photographer)
  studio_name VARCHAR(255),              -- Photographer's studio/business name
  specialty VARCHAR(255),                -- Specialty of the photographer (e.g., portrait, wedding)
  pricing TEXT,                     -- Details of pricing for photography services
  portfolio_url VARCHAR(255),             -- Link to photographer's portfolio
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- Profile creation timestamp
  FOREIGN KEY (user_id) REFERENCES users(id)  -- Foreign key to the user table (photographer)
);

-- Index for searching photographers by user_id
CREATE INDEX idx_profile_user ON photographer_profiles(user_id);
```

---

7. Review & Rating Table (reviews)

```
CREATE TABLE reviews (
  id INT AUTO_INCREMENT PRIMARY KEY,       -- Unique review identifier
  reviewer_id INT NOT NULL,             -- Foreign key referring to users.id (User giving the review)
  reviewee_id INT NOT NULL,             -- Foreign key referring to users.id (User being reviewed)
  rating INT NOT NULL,                -- Rating given (1 to 5 stars)
  comment TEXT,                    -- Comment content from reviewer
```

```sql
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, -- Timestamp when the review was posted

   FOREIGN KEY (reviewer_id) REFERENCES users(id), -- Foreign key for reviewer

   FOREIGN KEY (reviewee_id) REFERENCES users(id) -- Foreign key for reviewee

);


-- Index for searching reviews by reviewer or reviewee

CREATE INDEX idx_review_reviewer ON reviews(reviewer_id);

CREATE INDEX idx_review_reviewee ON reviews(reviewee_id);
```