# PhotoBazaar

Online artworks sharing & trading platform

Chang Liu

Xiaoxing Pan

Shiyuan Xu

https://photobazaar.vercel.app

# Background

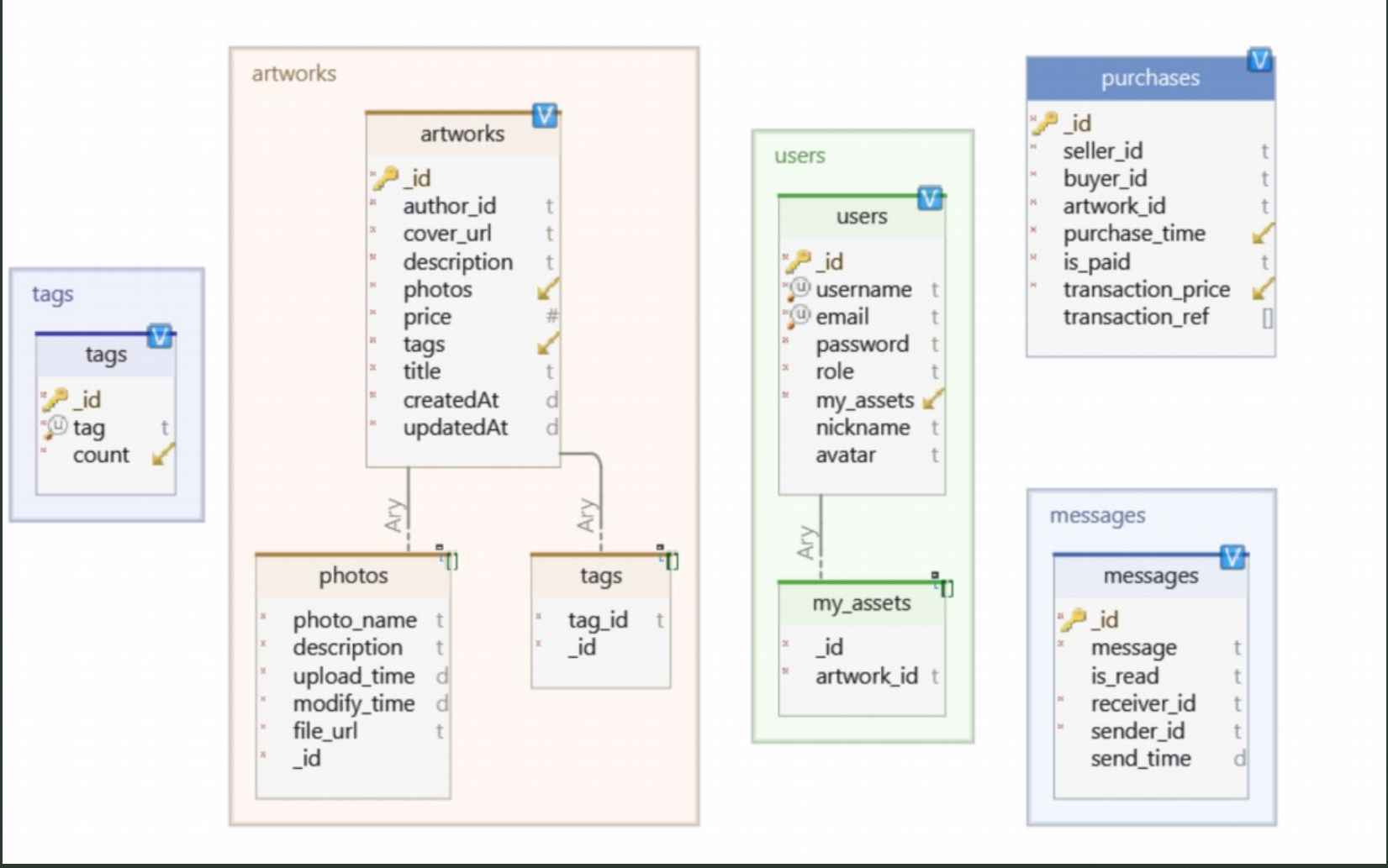
**PhotoBazaar** focuses on creating an Online artwork Sharing and Trading Platform, a digital marketplace where users can share, manage, and trade their original artworks.

- ✓ Author – Uploading & Management
- ✓ Buyer – Search, Cart, Payment
- ✓ Communication – Text Chat
- ✓ Admin – Tags, Users, Artworks, Payment Track

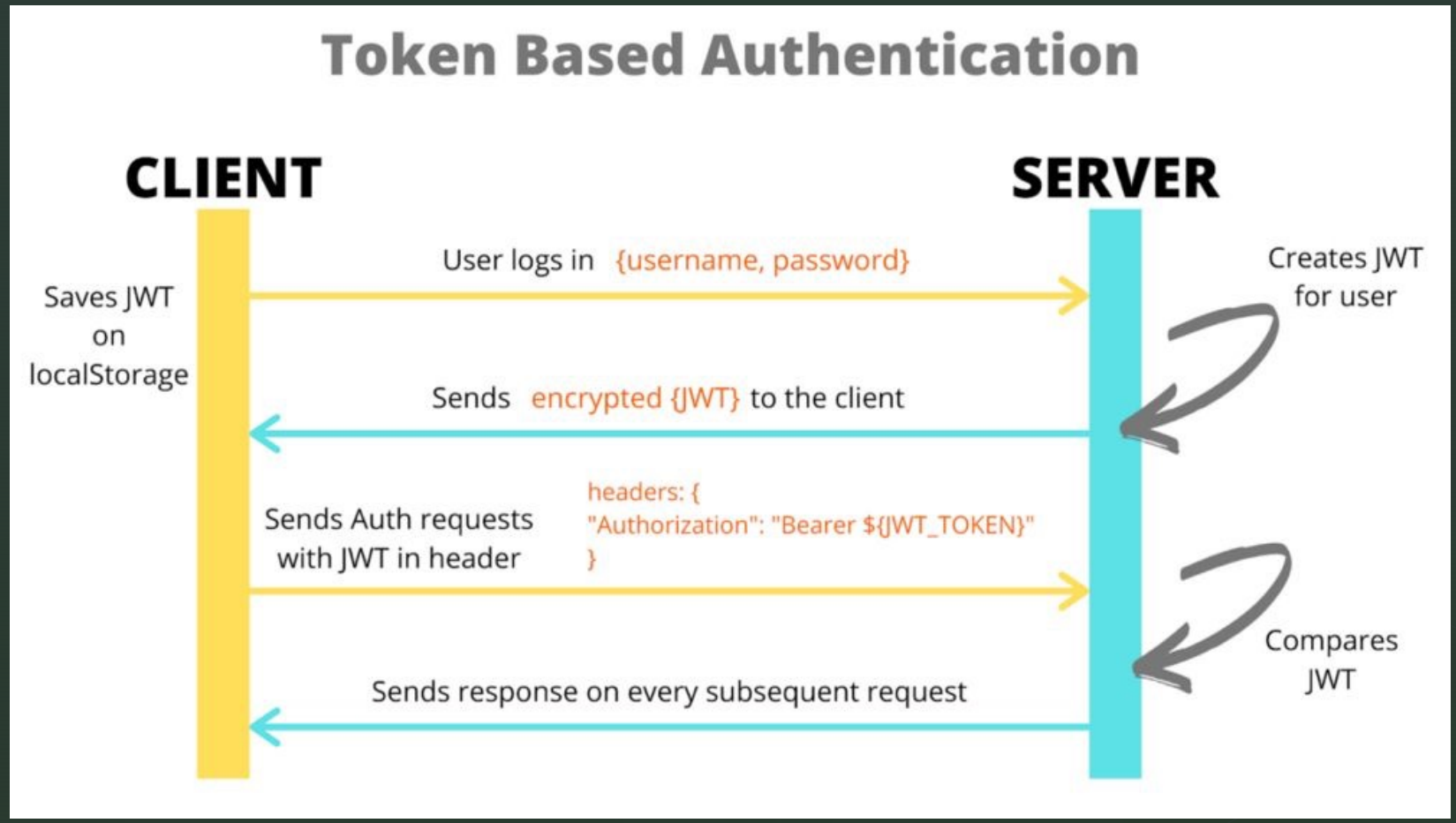
# Technologies

- ✓ MERN(MongoDB, Express, React, Node.js)
- ✓ Tailwind
- ✓ Jsonwebtoken, CORS, Axios
- ✓ AWS S3 Bucket
- ✓ Stripe
- ✓ nodemailer
- ✓ Socket.io
- ✓ Vercel

# Database - MongoDB

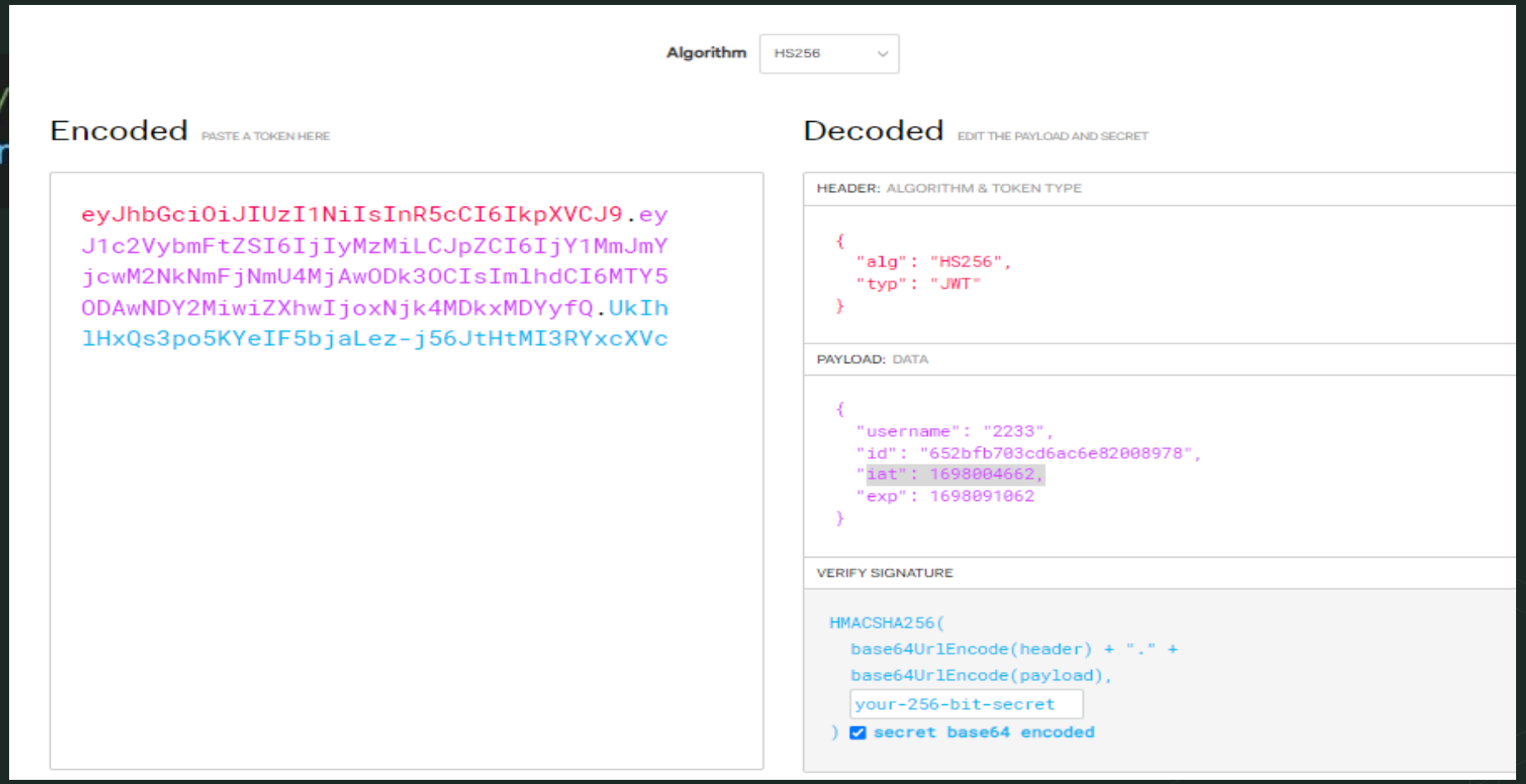


# Overview: Authentication&Authorization

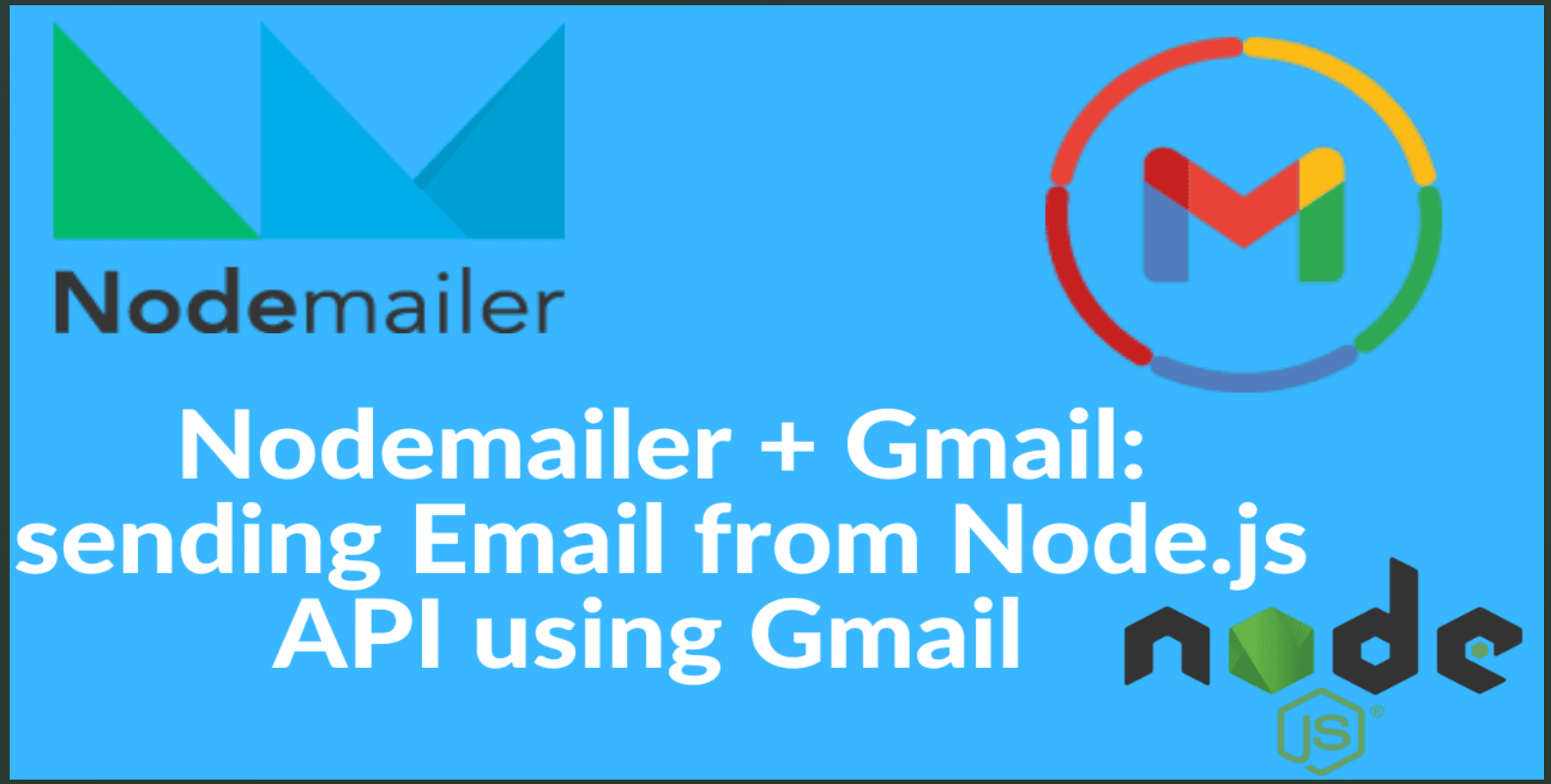


# Challenges & Solutions:

- ✓ JWT(Jsonwebtoken)



# Overview:
# Forgot Password & Reset Password



# Challenges & Solutions: [object File]

# What we learned

# Overview:

# Administration

# Challenges & Solutions:

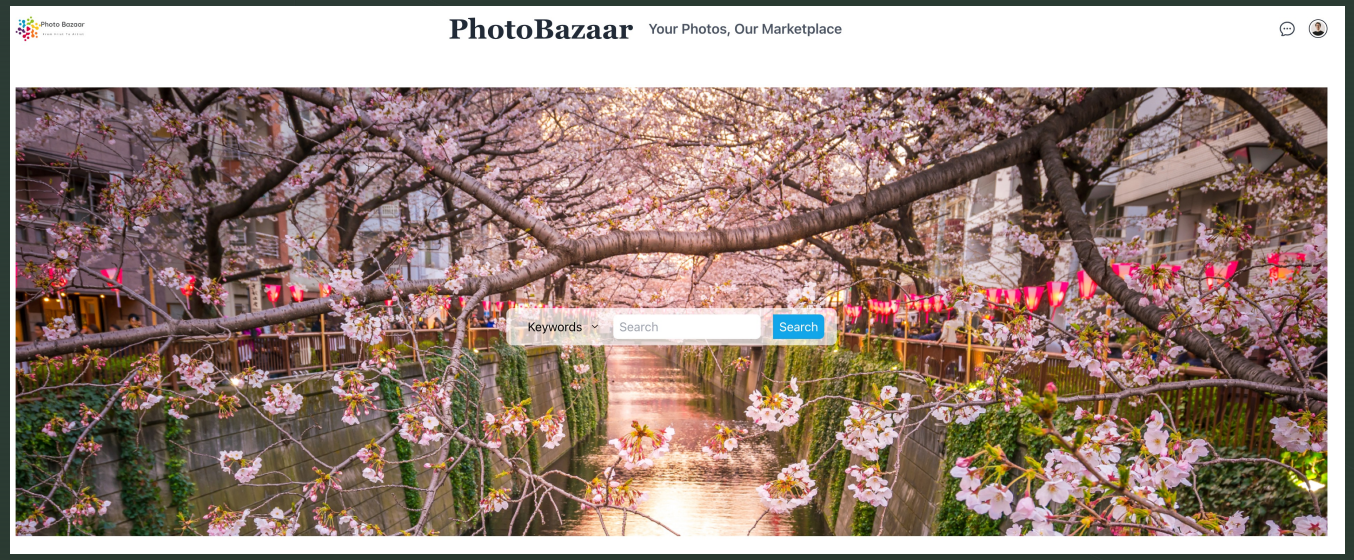
# What we learned

# Overview: Artworks

- ✓ Search => Tags/Keyword
- ✓ Sort => All/Price/Update Date
- ✓ Show => Home/My artworks/My assets
- ✓ Edit => CRUD artwork
- ✓ Details => CRUD photos
- Other => Add to Cart/Download

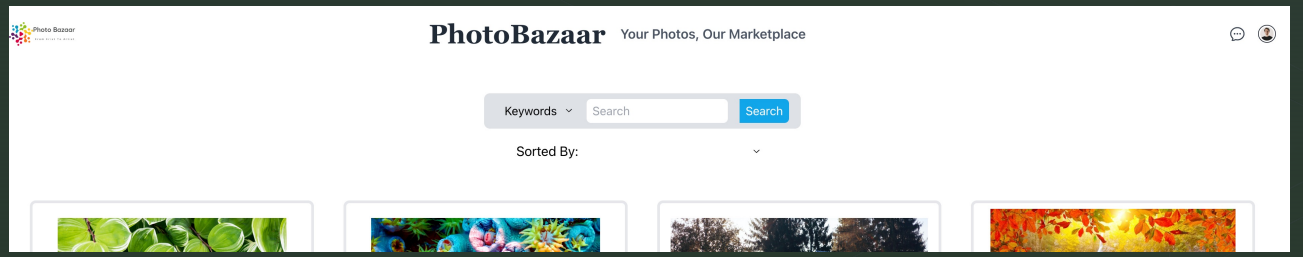
# Challenges & Solutions: Make component reusable

- Search Box => Home page/Search Page




Home Page

Search Page



- Search Box => Home page/Search Page

SearchBox Component

```
function SearchBoxComponent({ page }) {
```

```
  return (
    <div
      className={`${page === "search"
        ? "flex justify-center mt-5"
        : "absolute top-1/3 left-1/2 transform -translate-x-1/2 -translate-y-1/2"
        } p-4 searchBox`}
    >
      <div
        className={`${page === "search"
          ? "bg-gray-300 bg-opacity-70"
          : "bg-white bg-opacity-70"
          } rounded-lg p-2`}
```

Home Page

```
return (
  <div>
    <Header />
    <div className="search p-3 m-5">
      <img className="w-full" src="./homepagebg.png" alt="photobazaar"></img>
      <SearchBoxComponent />
    </div>
```

Search Page

```
<div>
    {page === "search" ? (
        <><SearchBoxComponent page="search" /></>
    ) : (<></>)}
</div>
```

# Challenges & Solutions: Make component reusable

- ArtworkList => Home page/Search Page/My Artworks Page/ My assets Page

Home/Search Page

My Artwork Page

My Assets Page

- ArtworkList => Home Page/Search Page/My artworks Page/My Assets Page

Artwork List Component

```
{/* show/edit photos */}
<button
    className={`${page === "home" ||
        page === "myAssets" ||
        page === "search"
        ? "border-r-2 pr-1" : ""} items-center pl-1`}
    onClick={() => {
        if (userId === null || userId === undefined && user.id === undefined) {
            openLoginModal();
            navigate("/Login");
        } else {
            navigate(`/details/${artwork._id}`, {
                state: { page: page },
            });
        }
    }}
>
    <svg …
    </svg>
</button>

{page === "myAssets" ||
    page === "home" ||
    page === "search" ? (
    <></>
) : (
    <>
        {/* edit main info */}
        <button …
        </button>
    </>
)}
```

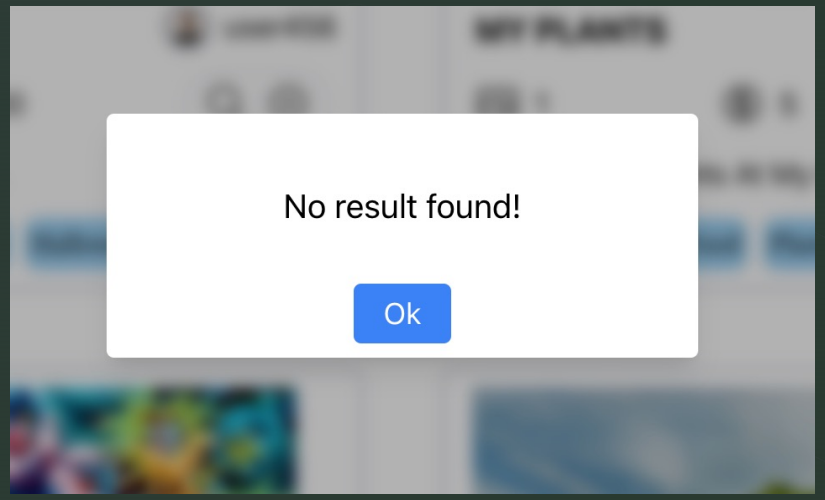
- AddArtwork => Add/Update

Artwork List Component

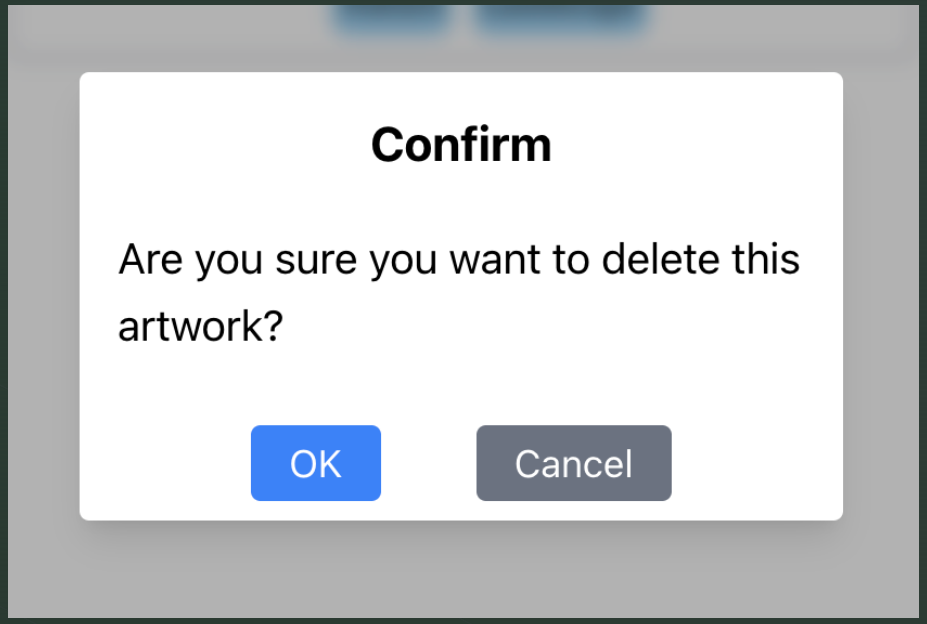
```
{/* show/edit photos */}
<button
    className={`${page === "home" ||
        page === "myAssets" ||
        page === "search"
        ? "border-r-2 pr-1" : ""} items-center pl-1`}
    onClick={() => {
        if (userId === null || userId === undefined && user.id === undefined) {
            openLoginModal();
            navigate("/Login");
        } else {
            navigate(`/details/${artwork._id}`, {
                state: { page: page },
            });
        }
    }}
>
    <svg …
    </svg>
</button>

{page === "myAssets" ||
    page === "home" ||
    page === "search" ? (
    <></>
) : (
    <>
        {/* edit main info */}
        <button …
        </button>
    </>
)}
```

- Modal




Warning/Information




Confirm

- Modal

```jsx
function Modal({ title, content, isOpen, onClick, proceed, confirm }) {
    return (
        isOpen && (
            <div className="modal fixed inset-0 flex bg-black bg-opacity-30 backdrop-blur-sm items-center justify-center z-50">
                <div className="bg-white p-2 rounded shadow-lg w-80">
                    <div className="modal-content p-2 ">
                        <div className="font-bold text-xl mb-5 flex justify-center">{title}</div>
                        <div className="text-lg flex justify-center">{content}</div>
                    </div>
                    <div className="modal-buttons flex justify-center  mt-5">
                        {confirm ? (<>
                            <button
                                className="px-4 py-1 mx-5 bg-blue-500  hover:bg-blue-600 rounded text-white"
                                onClick={proceed}>
                                OK
                            </button>
                        </>) : (<></>)}

                        <button
                            className={`${confirm ? "bg-gray-500 hover:bg-gray-600" : "bg-blue-500 hover:bg-blue-600"
                                } px-4 py-1 mx-5 text-white rounded`}
                            onClick={onClick}>
                            {confirm ? "Cancel" : "Ok"}
                        </button>
```

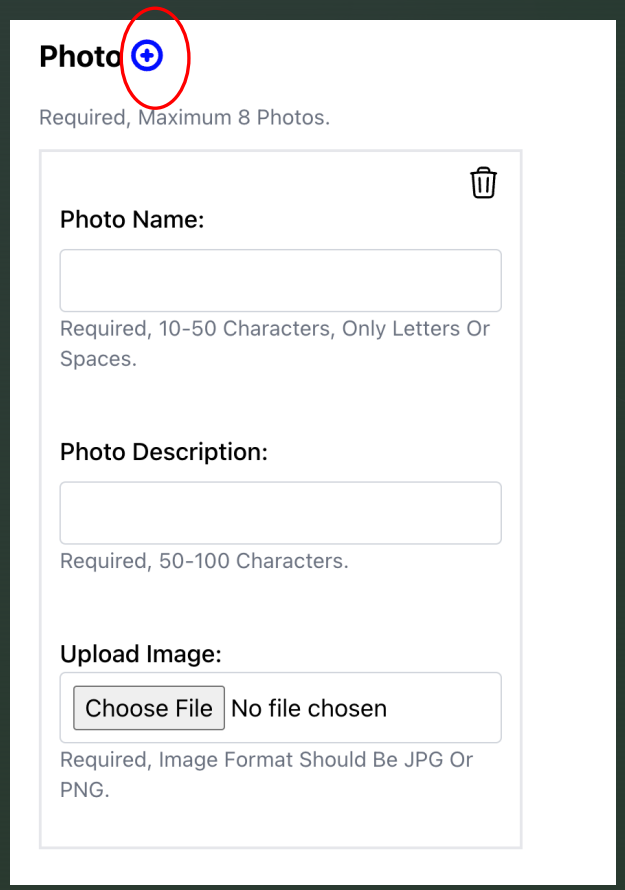
## Warning/Information

```jsx
<Modal
    title="Image Limit Exceeded"
    content="You can only upload 8 images!"
    onClick={closeImageLimitModal}
    isOpen={showImageLimitModal}
    alert={false}
/>
```

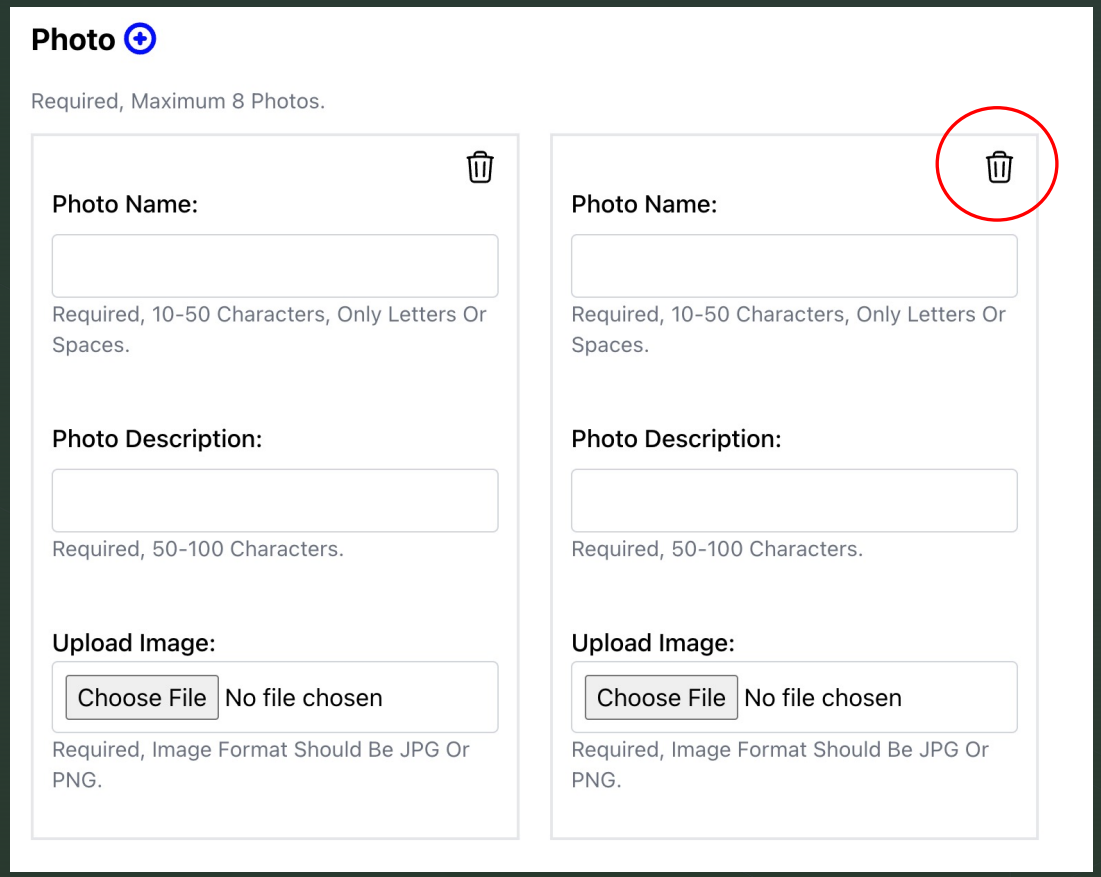
## Confirm

```jsx
<Modal
    title="Confirm"
    content="Are you sure you want to delete this artwork?"
    onClick={closeDeleteConfirmModal}
    isOpen={showDeleteConfirmModal}
    proceed={() => {
        deleteArtwork(artworkToDelete);
        closeDeleteConfirmModal();
    }}
    confirm={true}
/>
```

# Challenges & Solutions: Parent Component <=> Child Component




<=>



# Challenges & Solutions: Upload Pictures to S3

- Search Box => Home page/Search Page

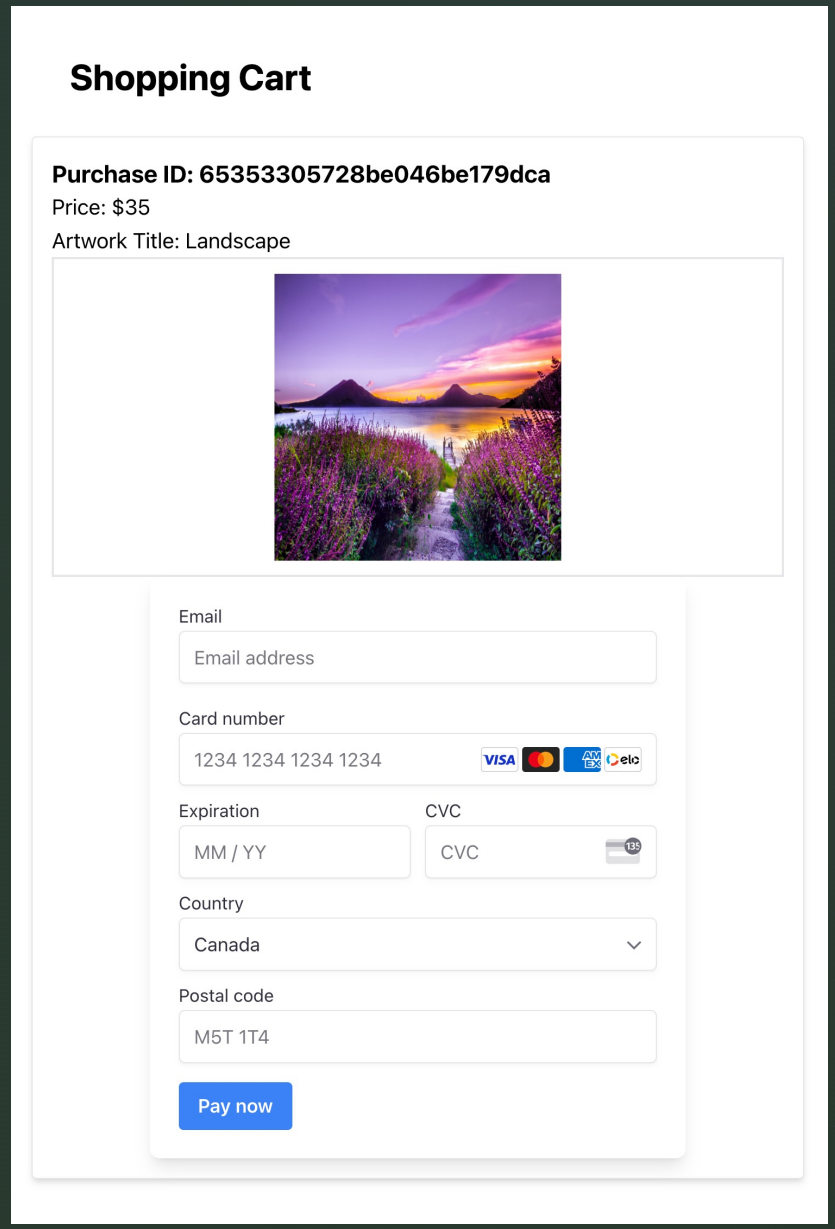
Home Page

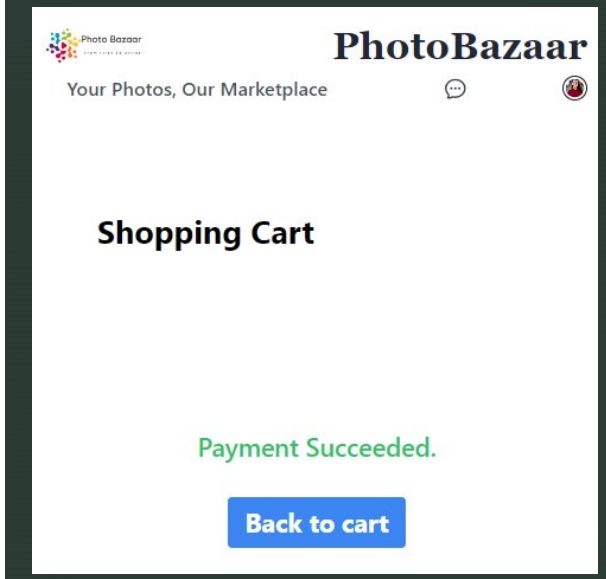
Search Page

# [object File]

# Overview:

# Payment



# Challenges & Solutions:

## Security & Compliance

- Stripe
- Stripe Checkout

## Payment Tracking

- Webhooks
- APIs

```
const { error } = await stripe.confirmPayment({
  elements,
  confirmParams: {
    return_url: deploy_api_url + `/payment_result/${purchase_id}`,
  },
});
```

# Challenges & Solutions:

Further chanllenge: how to obtain comprehensive information

- Payment intent ID
- Retry for failed payment
- Meta Data
- Database updating

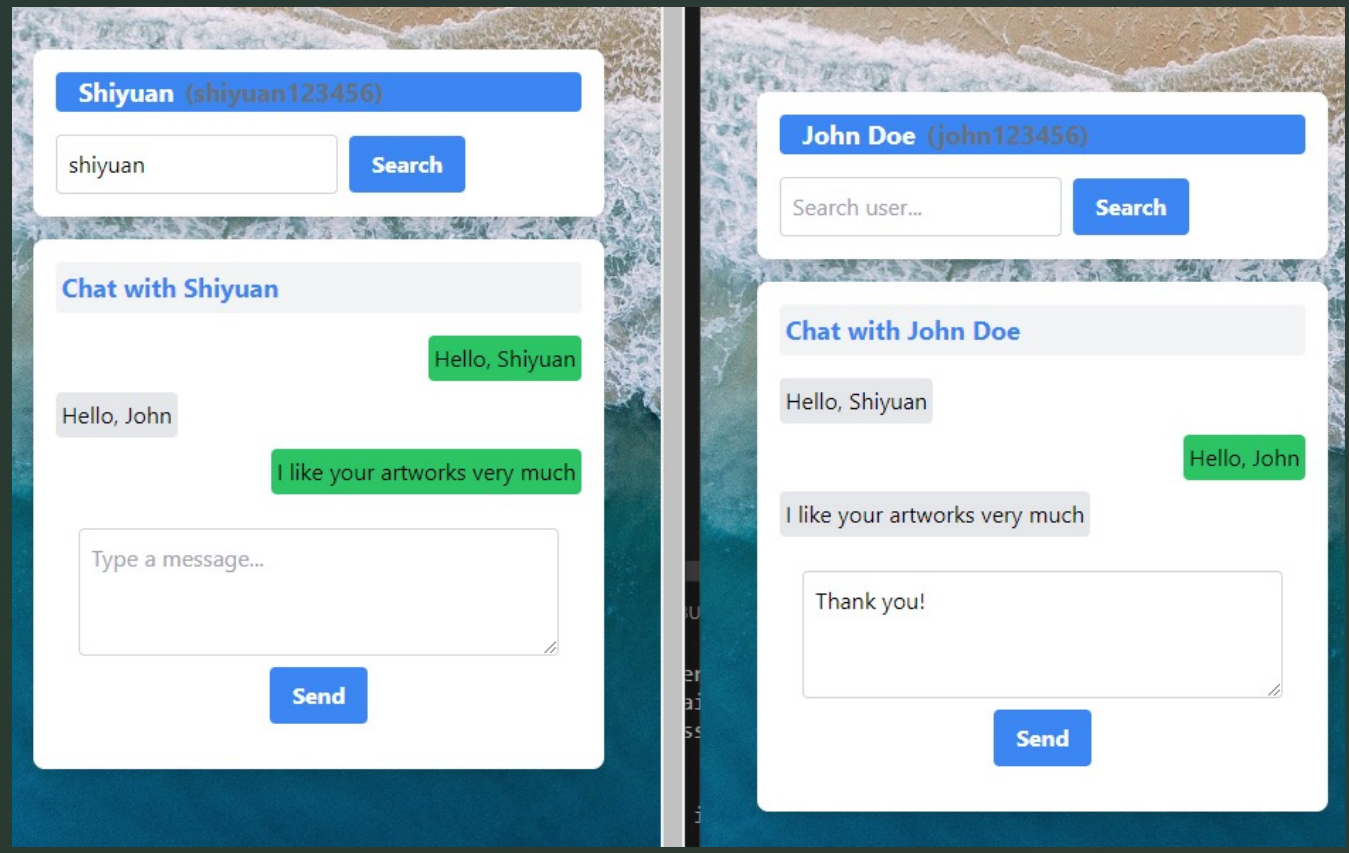
```
try {
    paymentIntent = await stripe.paymentIntents.create({
        amount: purchase.transaction_price * 100,
        currency: 'cad',
        description: purchase_id,
        metadata: { purchase_id: purchase_id },
        automatic_payment_methods: {
            enabled: true,
        },
    });
    const updatedTransactionRef = purchase.transaction_ref || [];
    updatedTransactionRef.push(paymentIntent.id);
    await PurchaseModel.updatePurchase(purchase_id, { transaction_ref: updatedTransactionRef });
    return res.status(200).json({ clientSecret: paymentIntent.client_secret, });
} catch (err) {
```

# What we learned

- Read the documentation carefully
- Documentation is not everything

# Overview:

# Communication




# Challenges & Solutions:

## Real-time chat

- Third-party APIs & SDK
  - MirrorFly, Rocket.Chat, Twilio, Apphitect …
- WebSocket protocol
- Socket.io

## Databse synchronization

- Message History
- Reading Status
- Notification

# Challenges & Solutions:

```
try {
  const insertedId = await Message.save(sender_id, receiver_id, message);
  const receiverSocket = userSockets[data.receiver_id];
  if (receiverSocket) {
    const sendback = data;
    sendback.id = insertedId;
    receiverSocket.emit('sendback', sendback);
  }       You, 2 weeks ago • move socket to controller
} catch (error) {
```

Integrate  accounting functions

Such as withdrawal, refund and report

# Future Work


# Summary