

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.

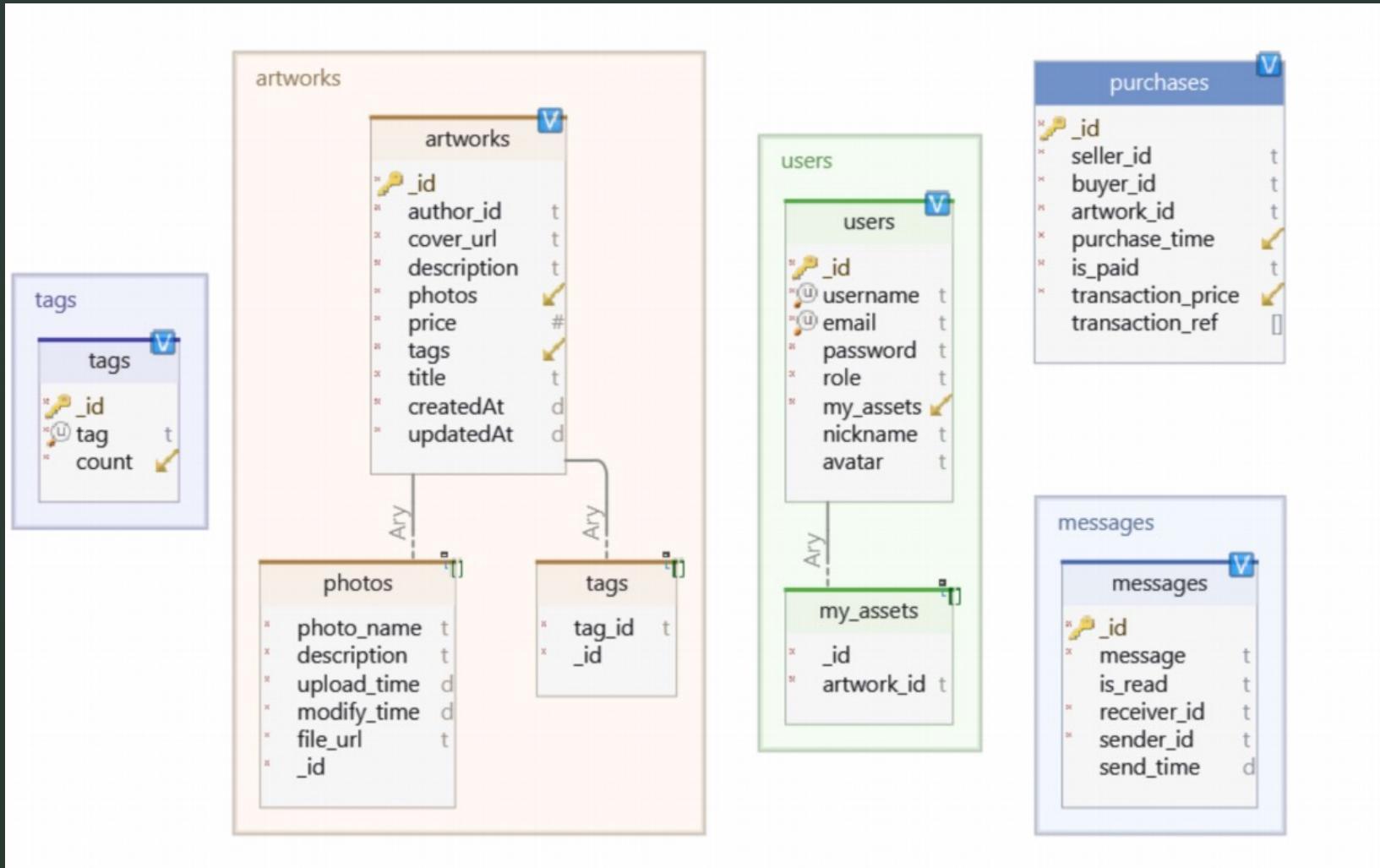
- ✓ Author – Uploading & Management
- ✓ Buyer – Search, Cart, Payment
- ✓ Communication – Text Chat
- ✓ Admin – Tags, Users, Artworks, Payment Track



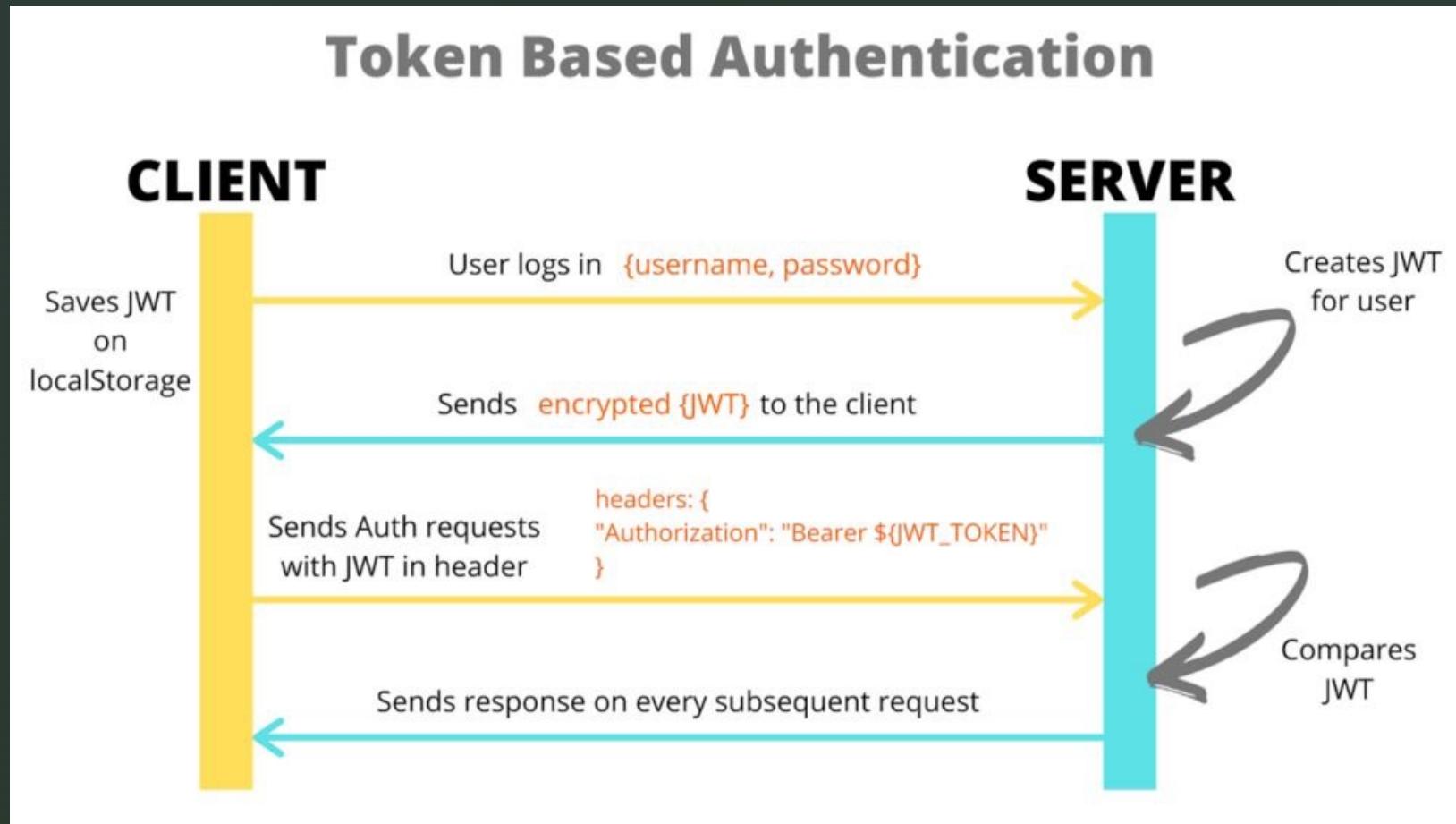
Technologies

- ✓ 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)

The screenshot shows the jwt.io interface. On the left, under 'Encoded' (PASTE A TOKEN HERE), is a long string of characters: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2Vyb... (truncated). On the right, under 'Decoded' (EDIT THE PAYLOAD AND SECRET), the token is broken down into its components. The 'HEADER: ALGORITHM & TOKEN TYPE' section shows a JSON object with 'alg': 'HS256' and 'typ': 'JWT'. The 'PAYLOAD: DATA' section shows a JSON object with 'username': '2233', 'id': '652fbf703cd6ac6e82008978', 'iat': 1698004662, and 'exp': 1698091062. The 'VERIFY SIGNATURE' section contains code for generating a HMACSHA256 signature using the header, payload, and a secret key ('your-256-bit-secret').

Algorithm HS256

Encoded PASTE A TOKEN HERE

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYOUT: DATA

```
{  
  "username": "2233",  
  "id": "652fbf703cd6ac6e82008978",  
  "iat": 1698004662,  
  "exp": 1698091062  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
)  secret base64 encoded
```

Overview: Forgot Password & Reset Password

Nodemailer + Gmail:
sending Email from Node.js
API using Gmail

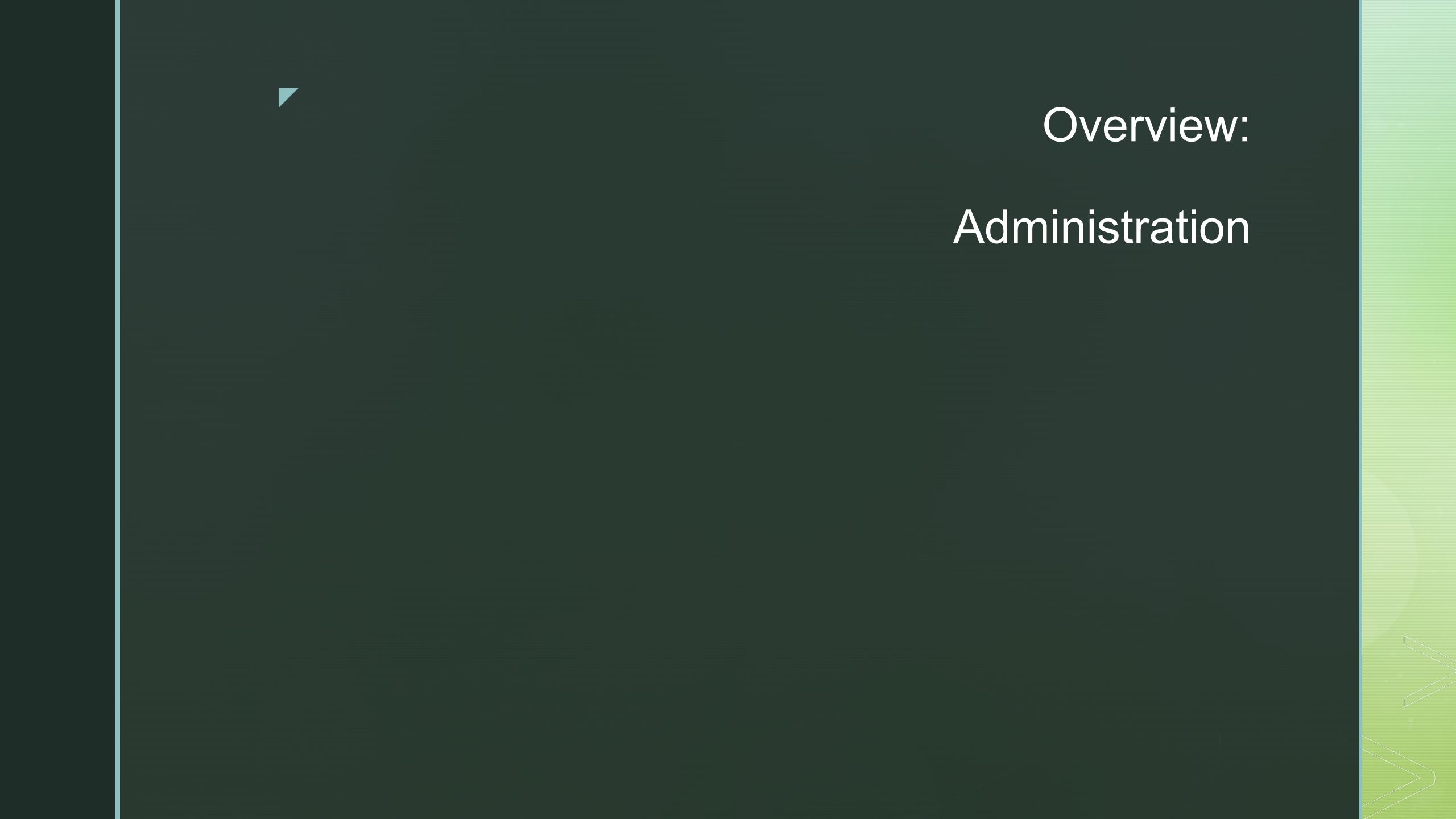
Challenges & Solutions: [object File]





What we learned





Overview: Administration

Challenges & Solutions:



What we learned

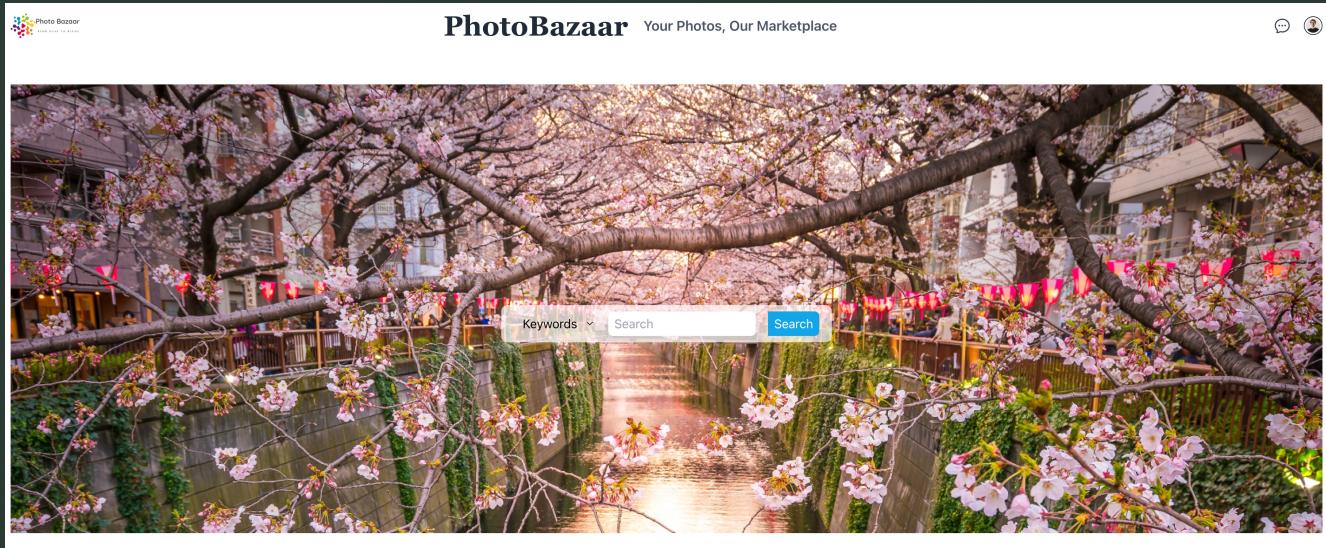


Overview: Artworks

- ✓ 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

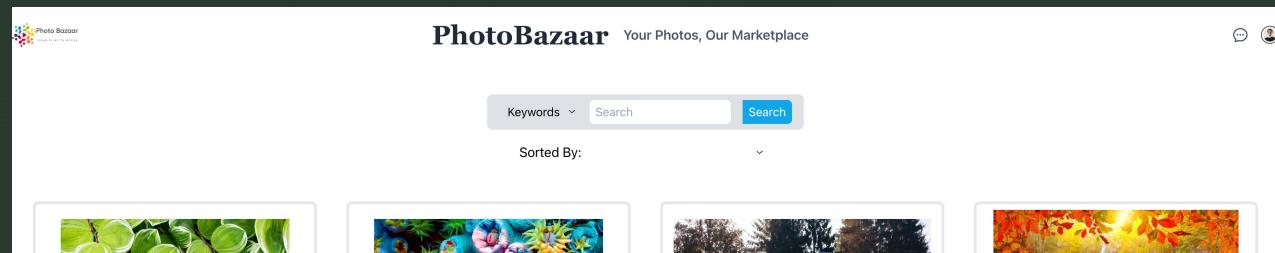
Challenges & Solutions: Make component reusable

- 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`}  
  )  
}
```

```
return (  
  <div>  
    <Header />  
    <div className="search p-3 m-5">  
      </img>  
      <SearchBoxComponent />  
    </div>  
  </div>
```

Home Page

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>
      </>
    </>
  )}
```

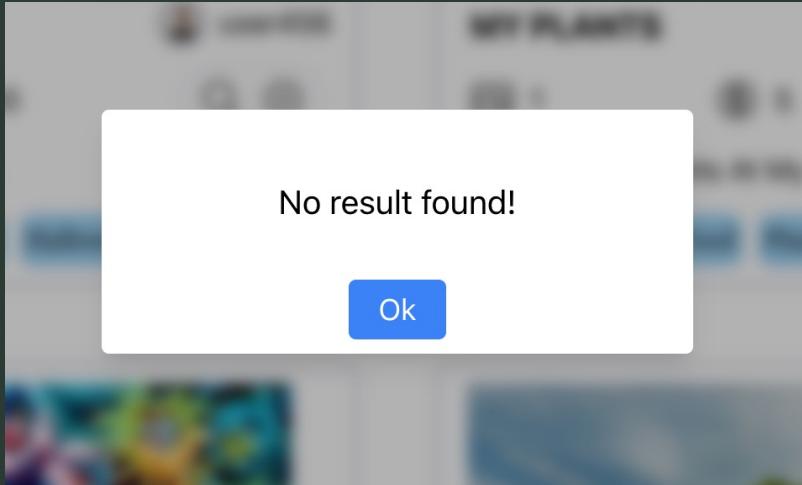
- AddArtwork => Add/Update

Artwork List Component

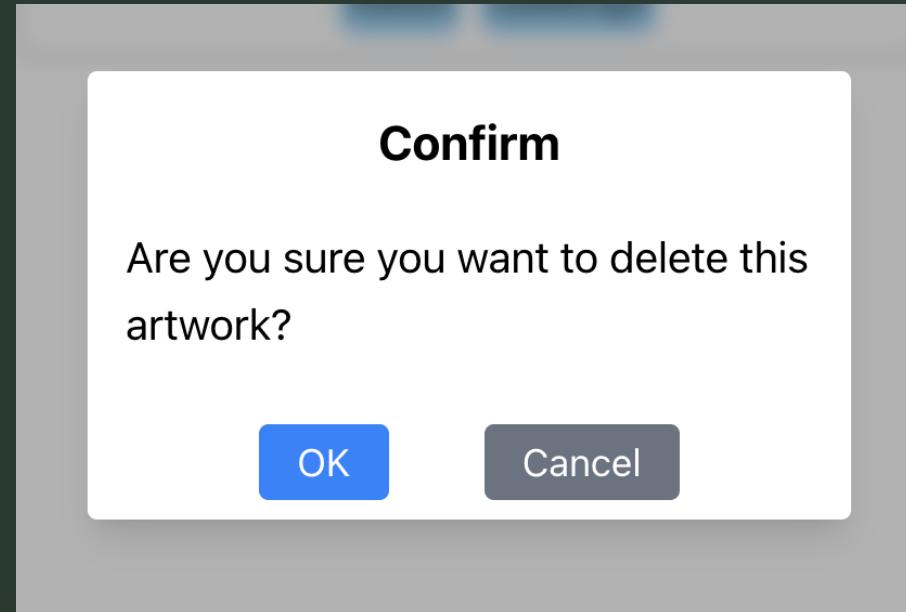
```
/* show/edit photos */
<button
  className={`${page === "home" ||
    page === "myAssets" ||
    page === "search"
    ? "border-r-2 pr-1" : ""} items-center pl-1`}
  onClick={() => {
    if (userId === null || userId === undefined && user.id === undefined) {
      openLoginModal();
      navigate("/Login");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

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

- Modal



Warning/Information



Confirm

■ Modal

```
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>
          </div>
        </div>
      </div>
    )
  );
}
```

Warning/Information

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

Confirm

```
<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

Photo

Required, Maximum 8 Photos.

Photo Name:

Required, 10-50 Characters, Only Letters Or Spaces.

Photo Description:

Required, 50-100 Characters.

Upload Image: No file chosen

Required, Image Format Should Be JPG Or PNG.

<=>

Photo

Required, Maximum 8 Photos.

Photo Name:

Required, 10-50 Characters, Only Letters Or Spaces.

Photo Description:

Required, 50-100 Characters.

Upload Image: No file chosen

Required, Image Format Should Be JPG Or PNG.

Photo Name:

Required, 10-50 Characters, Only Letters Or Spaces.

Photo Description:

Required, 50-100 Characters.

Upload Image: No file chosen

Required, Image Format Should Be JPG Or PNG.



Challenges & Solutions: Upload Pictures to S3

- Search Box => Home page/Search Page

Home Page

Search Page

[object File]

Overview: Payment

Shopping Cart

Purchase ID: 65353305728be046be179dca
Price: \$35
Artwork Title: Landscape



Email

Card number
1234 1234 1234 1234 

Expiration CVC

Country Canada

Postal code M5T 1T4

[Pay now](#)

Photo Bazaar
Your Photos, Our Marketplace 

Shopping Cart

Payment Succeeded.

[Back to cart](#)

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 challenge: how to obtain comprehensive information

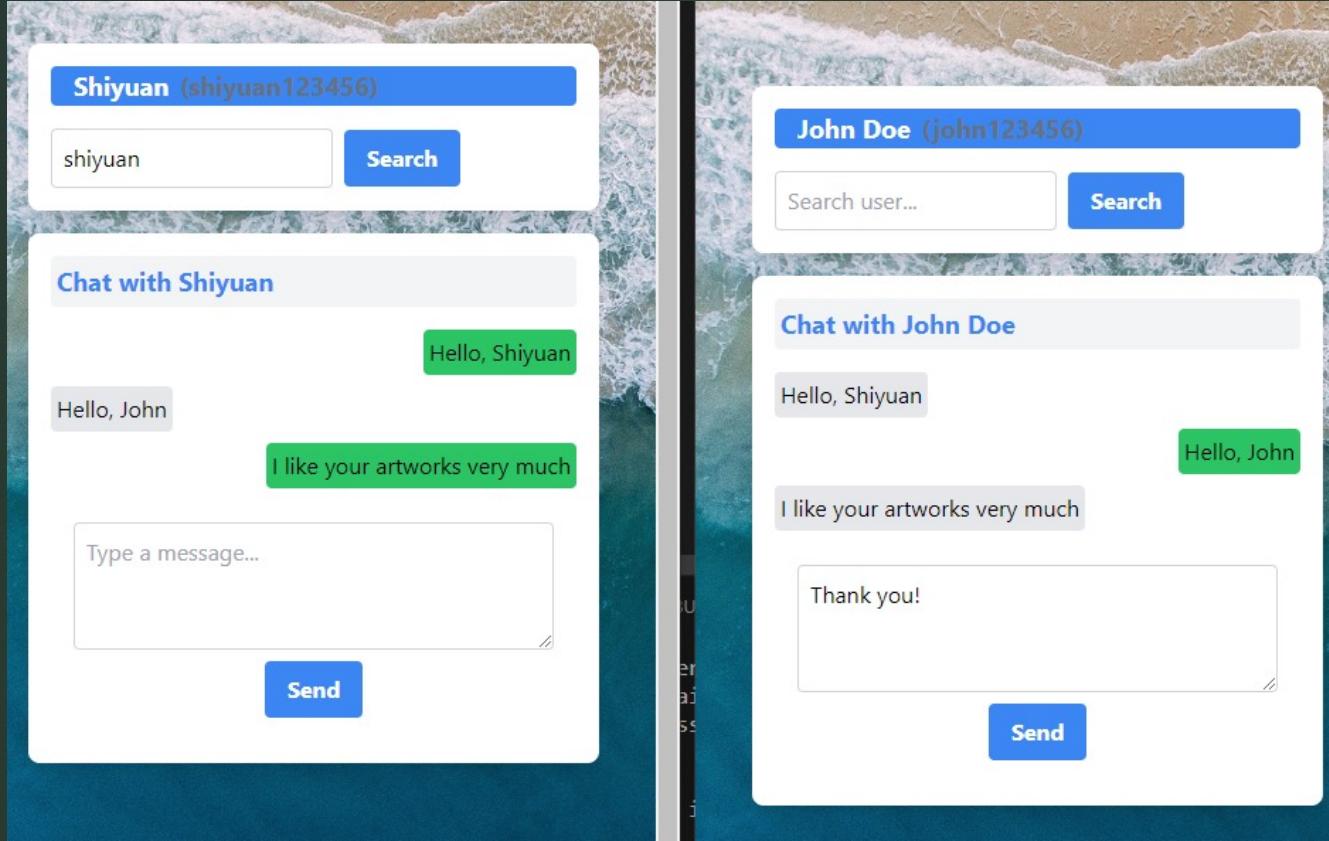
- Payment intent ID
- Retry for failed payment
- Meta Data
- Database updating

```
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

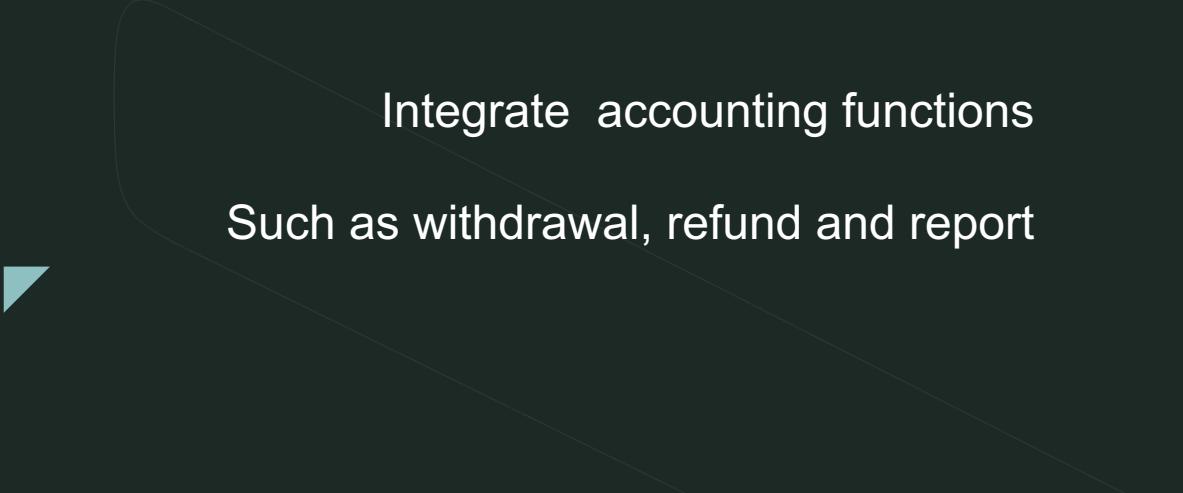
Database synchronization

- 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);
  }
} catch (error) {
```

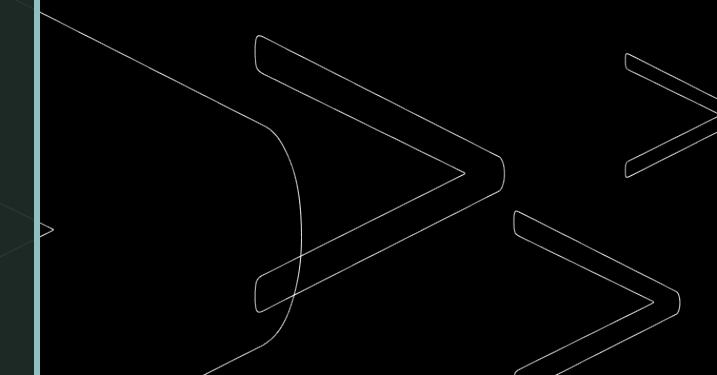
You, 2 weeks ago • move socket to controller



Integrate accounting functions

Such as withdrawal, refund and report

Future Work



Summary

