CLOTHO

Meredith White

Dustin Ruck

Soliloquy Yarrow

BACKGROUND

- Peer-to-peer marketplace app like the fashion platform Depop
- Buy and sell on the same platform
- Make purchases from multiple sellers

DEPOP FEATURES

- Complex searching
- Suggestions
- Promotions
- Product likes
- User follows

- Seller shops
- Seller reviews
- Purchasing options
    - Buy now or offers
- Payment options

CLOTHO FEATURES

Users:

- Sign up, log in/out
- Browse products
- Search and filter results
- Browse seller shops
- Place orders
- Post items for sale
- Manage profile

Admins:

- Log in/out
- Manage user accounts
- View charts
- View product tables

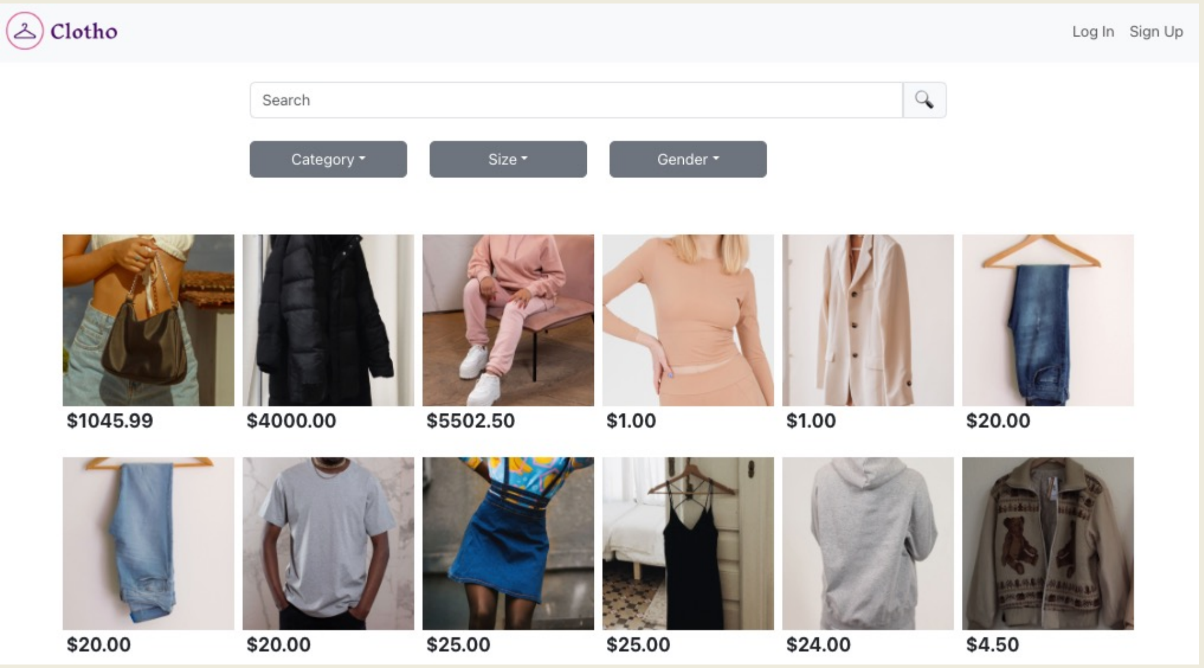
OUR APPROACH

Initial roles:

- Meredith: back end
- Soli: front end
- Dustin: cloud technologies

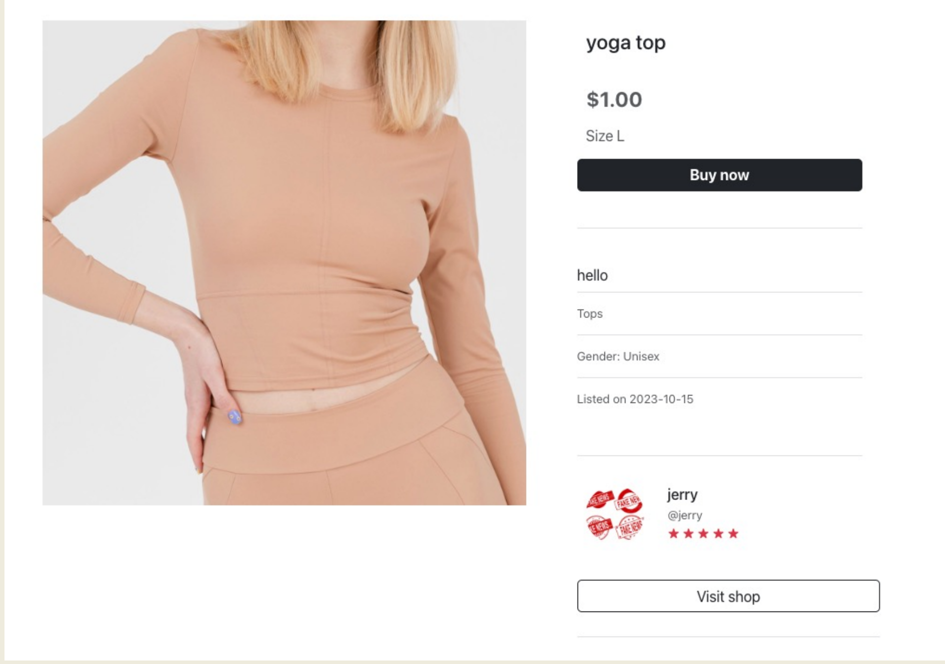
Later:

- Task rotation



$1045.99 $4000.00 $5502.50 $1.00 $1.00 $20.00

$20.00 $20.00 $25.00 $25.00 $24.00 $4.50


yoga top

$1.00

Size L

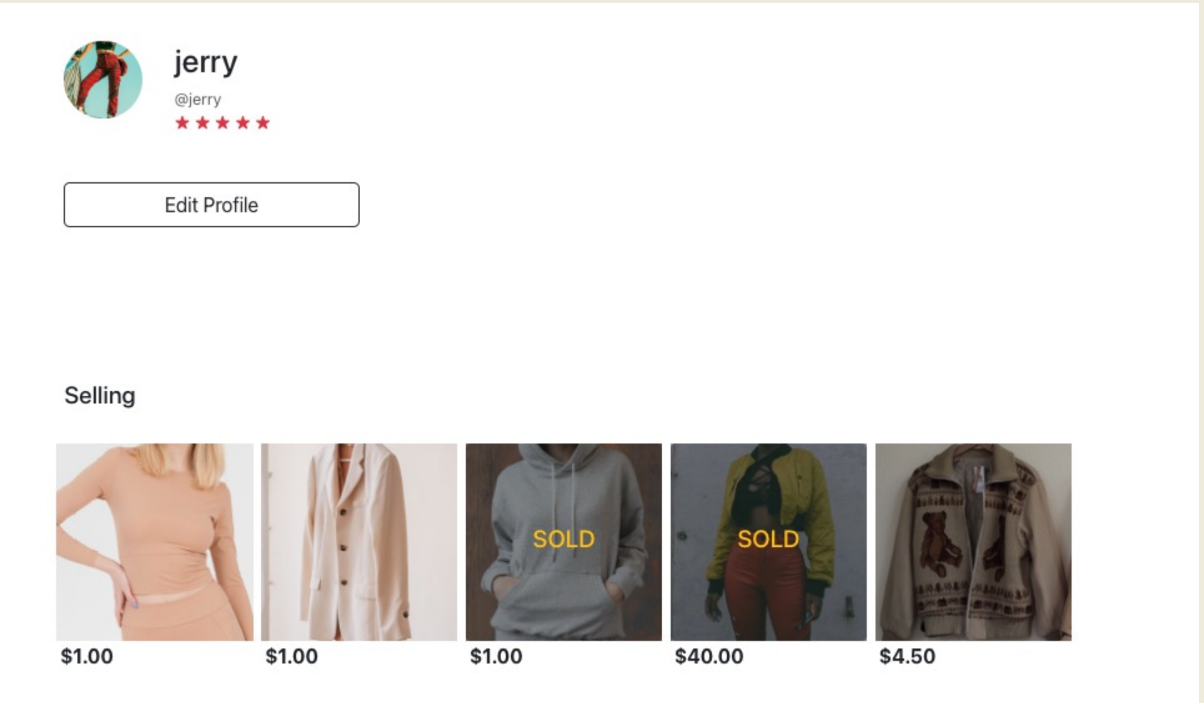
hello

Tops

Gender: Unisex

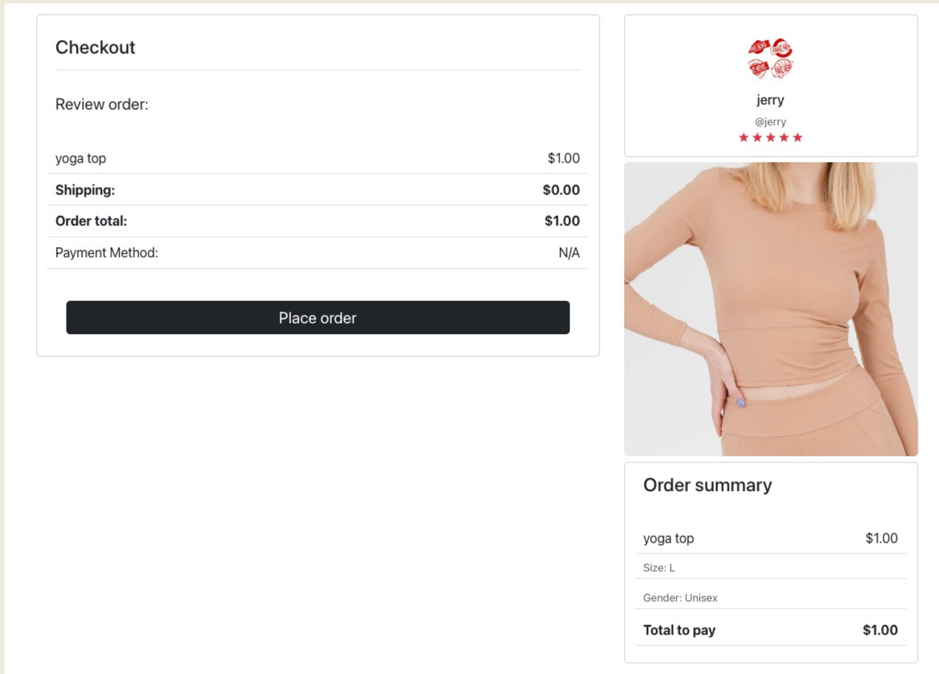
Listed on 2023-10-15

jerry
@jerry
★★★★★


jerry
@jerry
★★★★★

Selling

$1.00 $1.00 SOLD $1.00 SOLD $40.00 $4.50


Checkout

Review order:

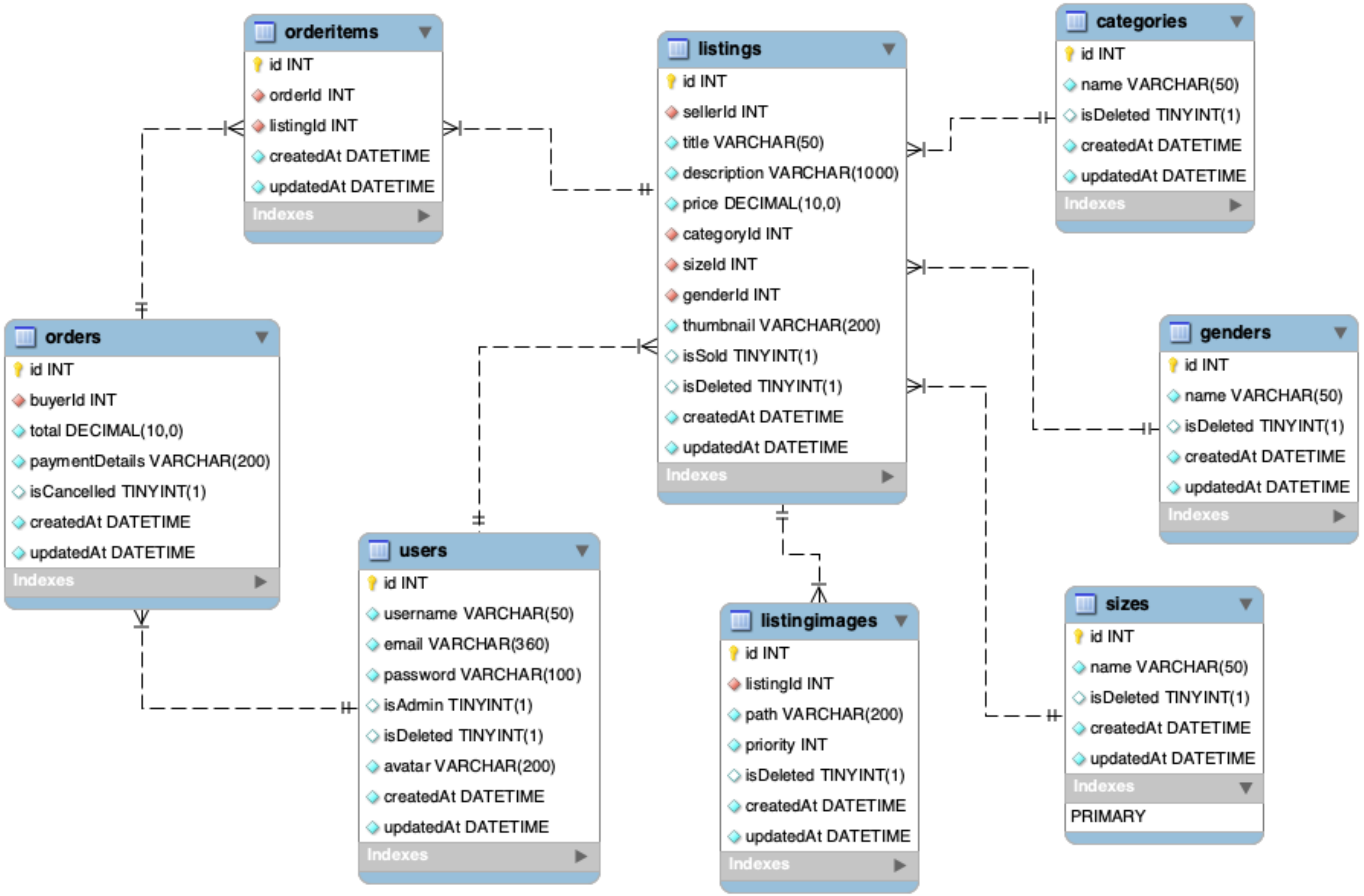
yoga top	$1.00
Shipping:	**$0.00**
Order total:	**$1.00**
Payment Method:	N/A

jerry
@jerry
★★★★★

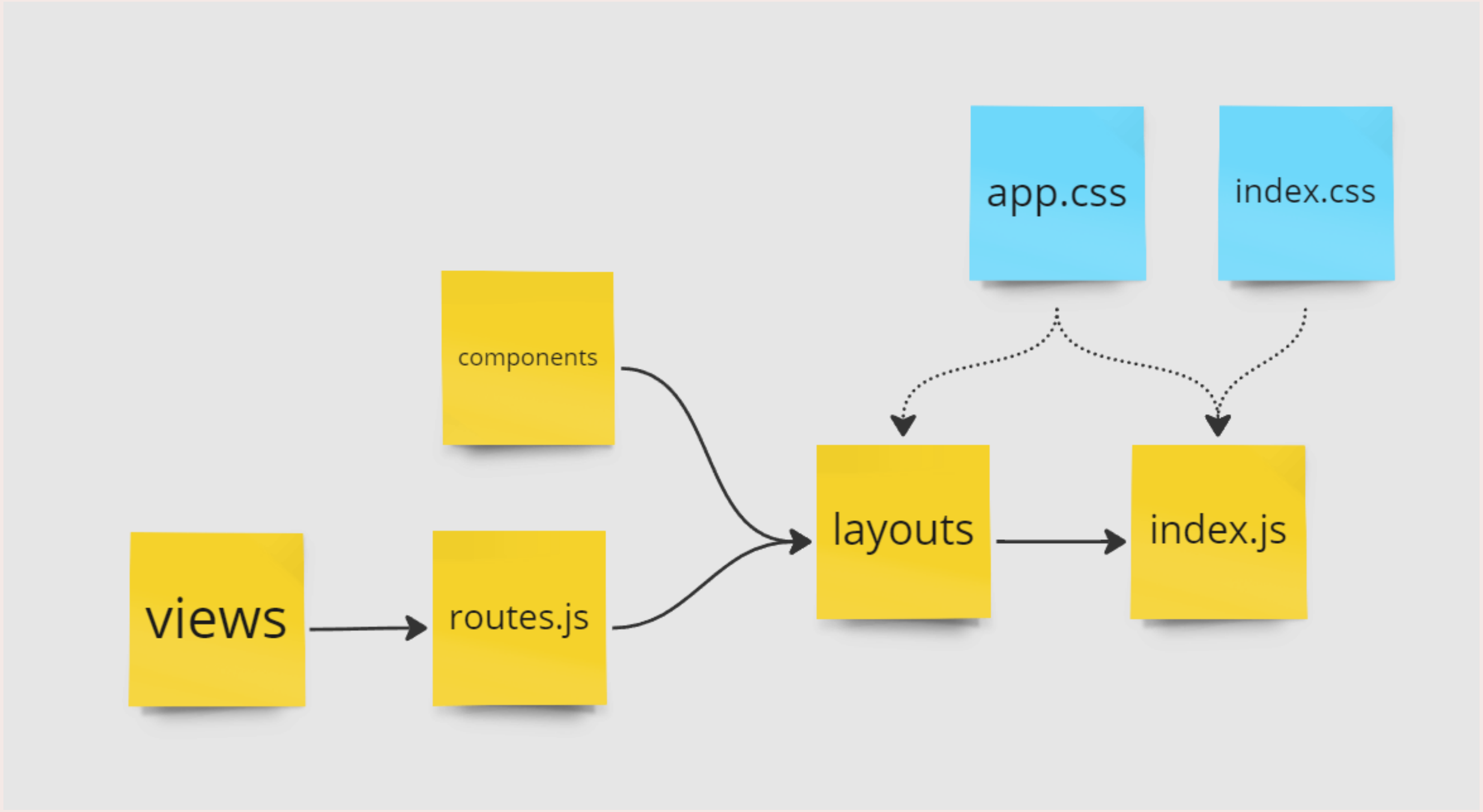
Order summary

yoga top	$1.00
Size: L	
Gender: Unisex	
Total to pay	**$1.00**

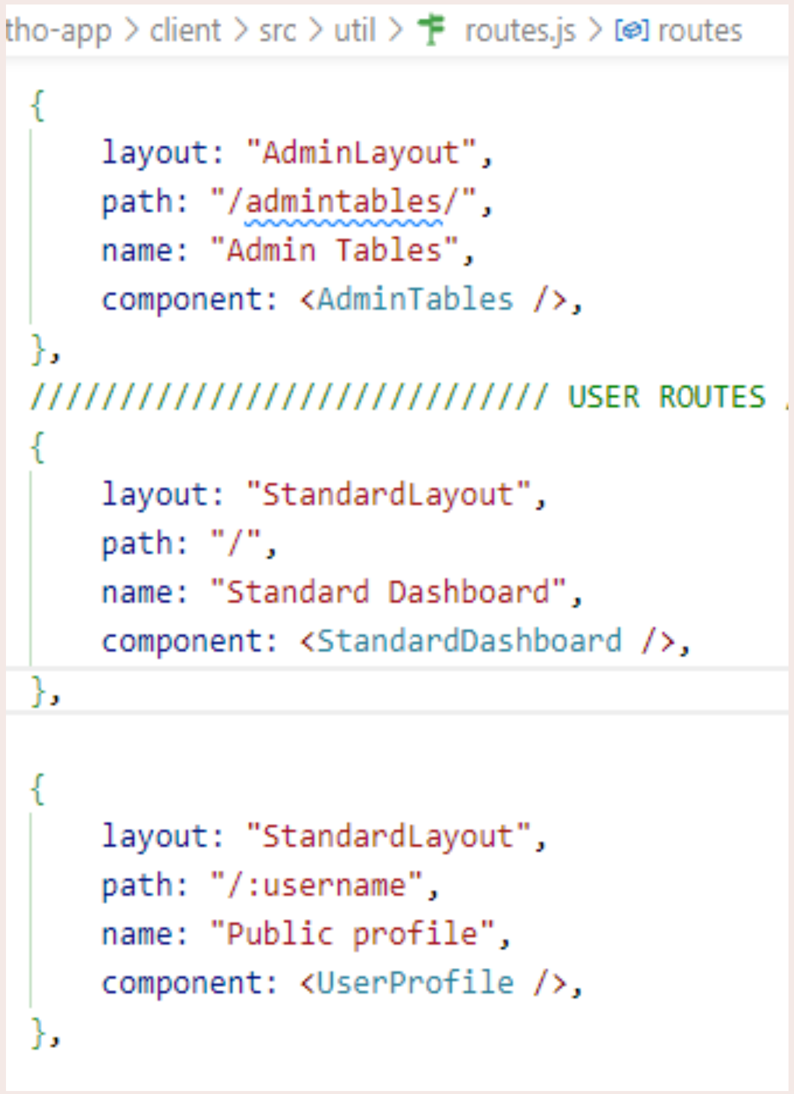
CHALLENGES & SOLUTIONS


orderitems
id INT
orderId INT
listingId INT
createdAt DATETIME
updatedAt DATETIME
Indexes
listings
id INT
sellerId INT
title VARCHAR(50)
description VARCHAR(1000)
price DECIMAL(10,0)
categoryId INT
sizeId INT
genderId INT
thumbnail VARCHAR(200)
isSold TINYINT(1)
isDeleted TINYINT(1)
createdAt DATETIME
updatedAt DATETIME
Indexes
categories
id INT
name VARCHAR(50)
isDeleted TINYINT(1)
createdAt DATETIME
updatedAt DATETIME
Indexes
orders
id INT
buyerId INT
total DECIMAL(10,0)
paymentDetails VARCHAR(200)
isCancelled TINYINT(1)
createdAt DATETIME
updatedAt DATETIME
Indexes
genders
id INT
name VARCHAR(50)
isDeleted TINYINT(1)
createdAt DATETIME
updatedAt DATETIME
Indexes
users
id INT
username VARCHAR(50)
email VARCHAR(360)
password VARCHAR(100)
isAdmin TINYINT(1)
isDeleted TINYINT(1)
avatar VARCHAR(200)
createdAt DATETIME
updatedAt DATETIME
Indexes
listingimages
id INT
listingId INT
path VARCHAR(200)
priority INT
isDeleted TINYINT(1)
createdAt DATETIME
updatedAt DATETIME
Indexes
sizes
id INT
name VARCHAR(50)
isDeleted TINYINT(1)
createdAt DATETIME
updatedAt DATETIME
Indexes
PRIMARY

CHALLENGE: PROPS & ROUTING



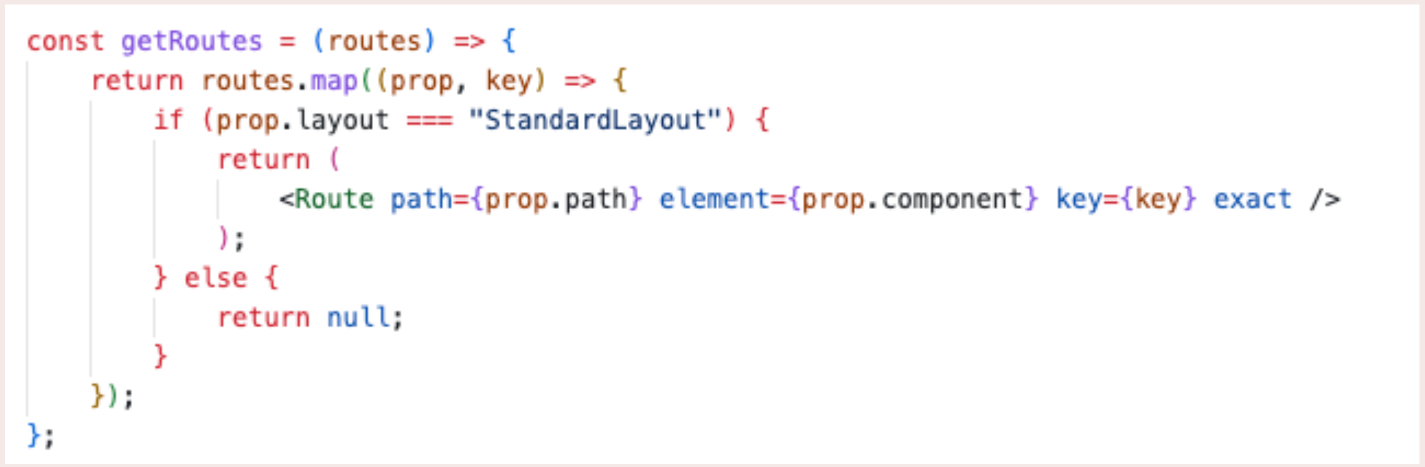
ROUTES.JS



tho-app > client > src > util > routes.js > routes

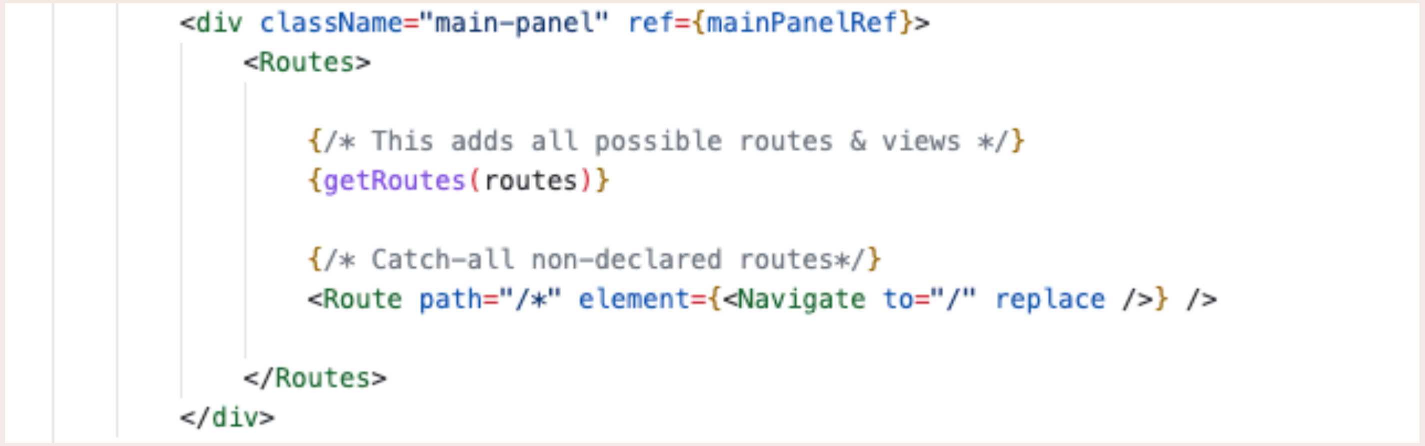
```
{
    layout: "AdminLayout",
    path: "/admintables/",
    name: "Admin Tables",
    component: <AdminTables />,
},
//////////////////////////// USER ROUTES
{
    layout: "StandardLayout",
    path: "/",
    name: "Standard Dashboard",
    component: <StandardDashboard />,
},

{
    layout: "StandardLayout",
    path: "/:username",
    name: "Public profile",
    component: <UserProfile />,
},
```

ROUTE MAP



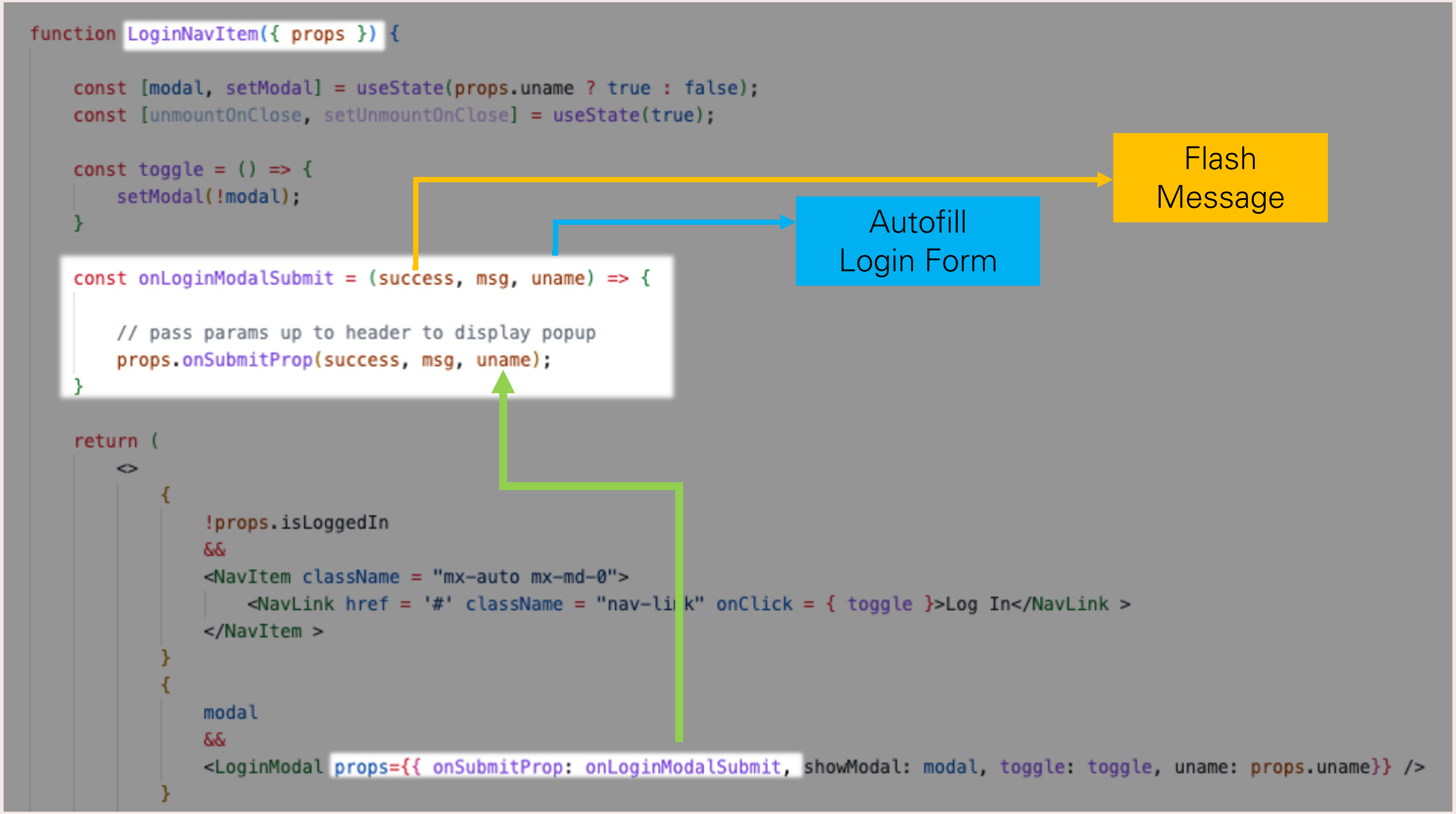
```
const getRoutes = (routes) => {
    return routes.map((prop, key) => {
        if (prop.layout === "StandardLayout") {
            return (
                <Route path={prop.path} element={prop.component} key={key} exact />
            );
        } else {
            return null;
        }
    });
};
```

LAYOUT



```
        <div className="main-panel" ref={mainPanelRef}>
            <Routes>

                {/* This adds all possible routes & views */}
                {getRoutes(routes)}

                {/* Catch-all non-declared routes*/}
                <Route path="/*" element={<Navigate to="/" replace />} />

            </Routes>
        </div>
```

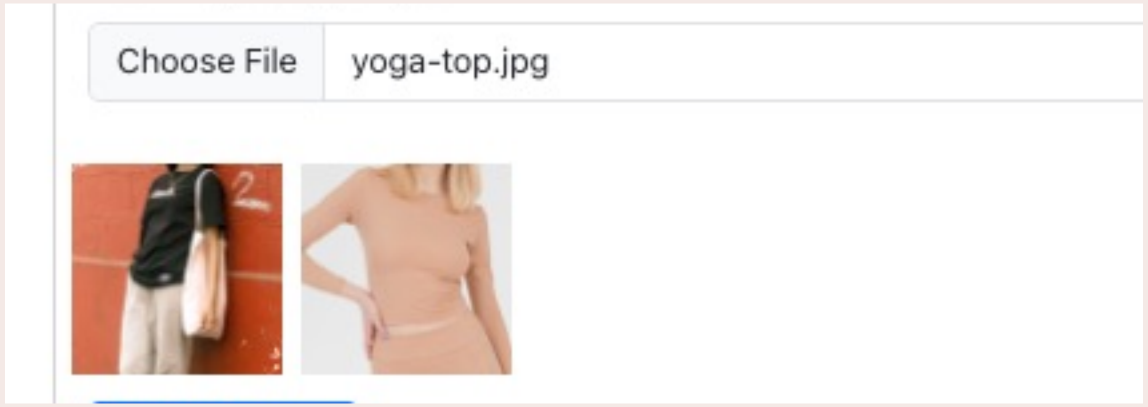
FUNCTIONS AS PROPS



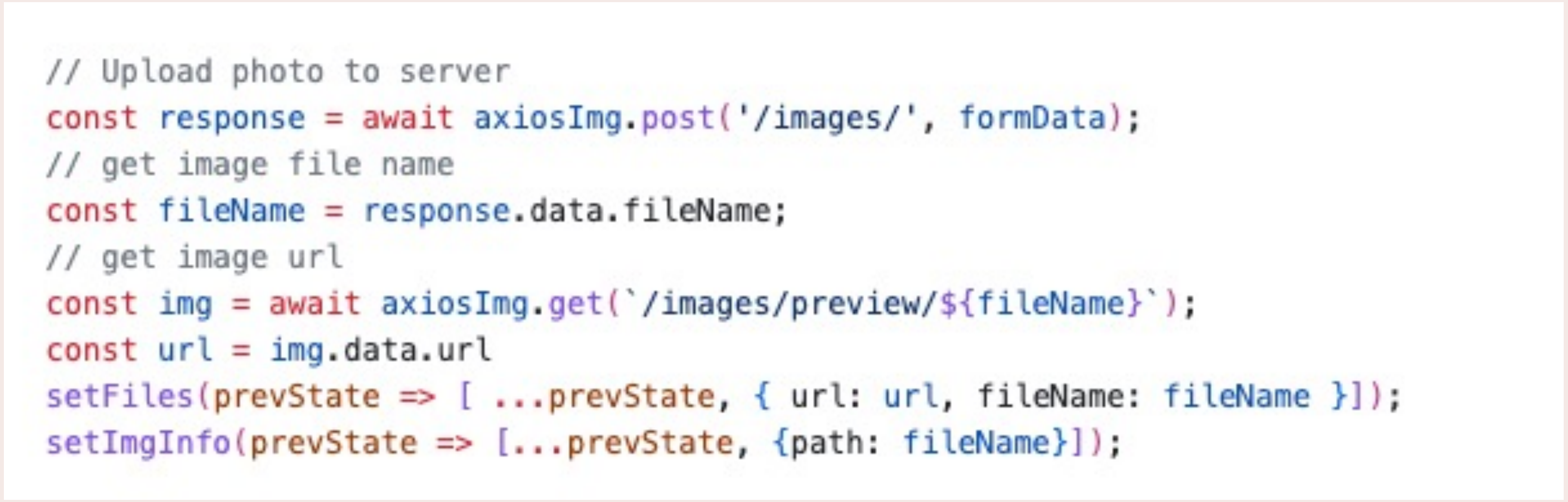
```
function LoginNavItem({ props }) {

    const [modal, setModal] = useState(props.uname ? true : false);
    const [unmountOnClose, setUnmountOnClose] = useState(true);

    const toggle = () => {
        setModal(!modal);
    }

    const onLoginModalSubmit = (success, msg, uname) => {

        // pass params up to header to display popup
        props.onSubmitProp(success, msg, uname);
    }

    return (
        <>
            {
                !props.isLoggedIn
                &&
                <NavItem className = "mx-auto mx-md-0">
                    <NavLink href = '#' className = "nav-link" onClick = { toggle }>Log In</NavLink >
                </NavItem >
            }
            {

                modal
                &&
                <LoginModal props={{ onSubmitProp: onLoginModalSubmit, showModal: modal, toggle: toggle, uname: props.uname}} />
            }
```

CHALLENGE: S3 IMAGES

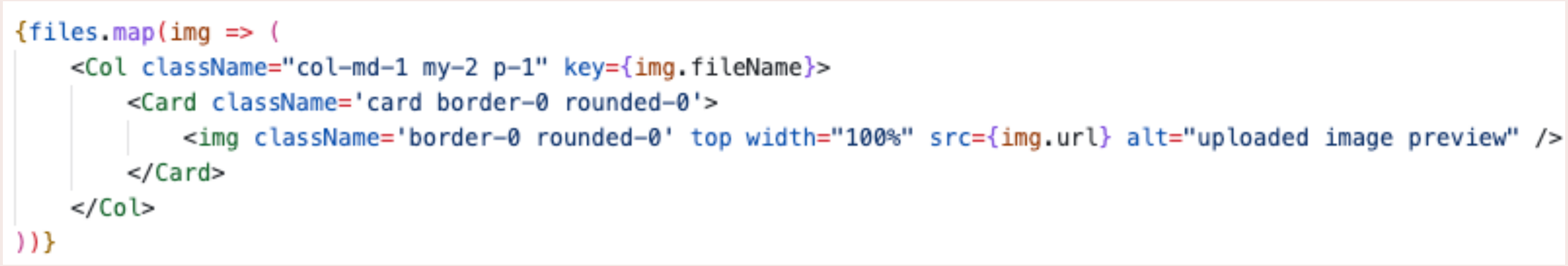
- Neither files nor URLs stored in database
- Saving and exposing images requires sequential requests



IMAGES SOLUTION



```javascript
// Upload photo to server
const response = await axiosImg.post('/images/', formData);
// get image file name
const fileName = response.data.fileName;
// get image url
const img = await axiosImg.get(`/images/preview/${fileName}`);
const url = img.data.url
setFiles(prevState => [ ...prevState, { url: url, fileName: fileName }]);
setImgInfo(prevState => [...prevState, {path: fileName}]);
```



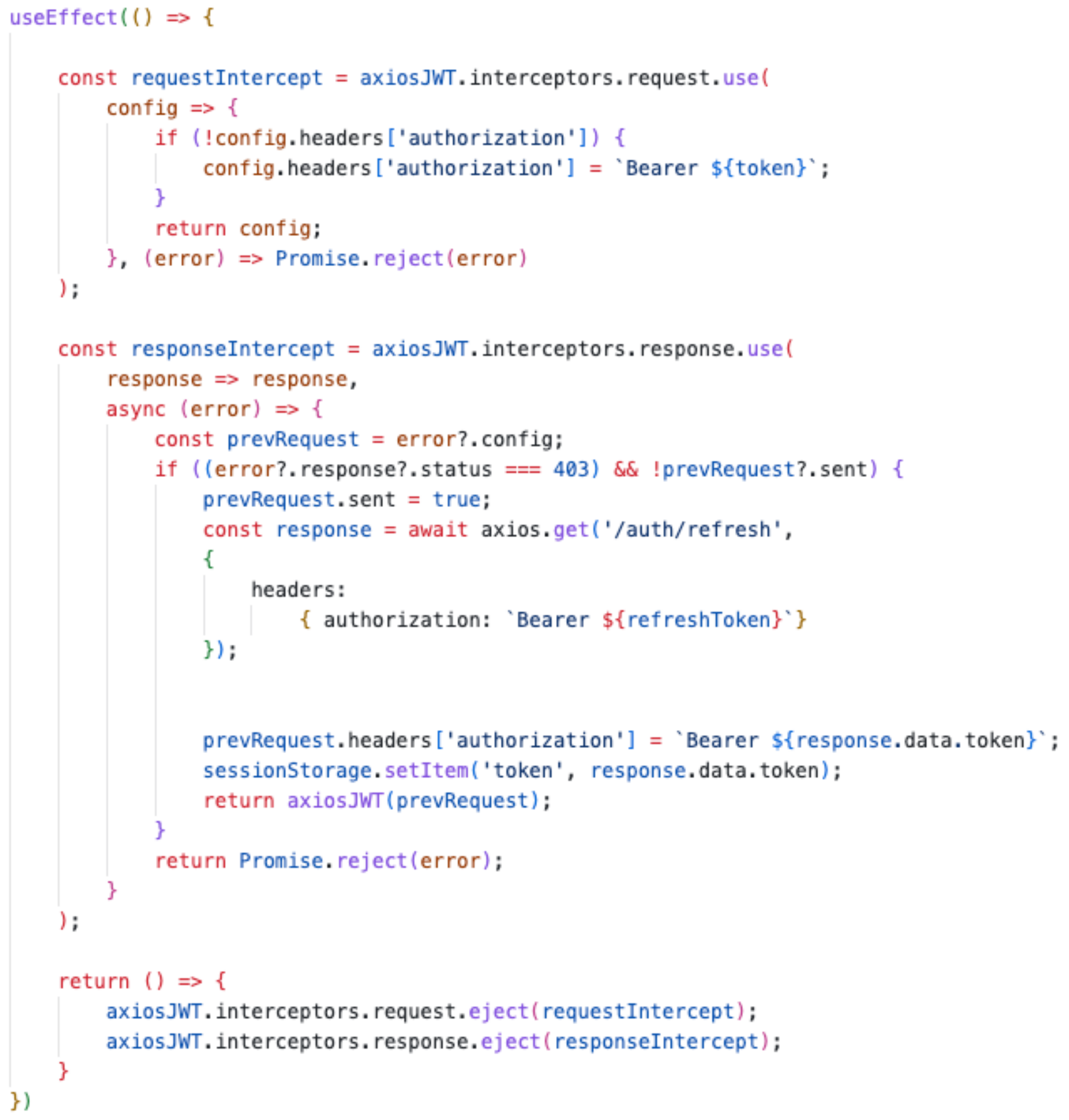
```jsx
{files.map(img => (
    <Col className="col-md-1 my-2 p-1" key={img.fileName}>
        <Card className='card border-0 rounded-0'>
            <img className='border-0 rounded-0' top width="100%" src={img.url} alt="uploaded image preview" />
        </Card>
    </Col>
))}
```

NEW LEARNING

MEREDITH: REQUEST & RESPONSE INTERCEPTS

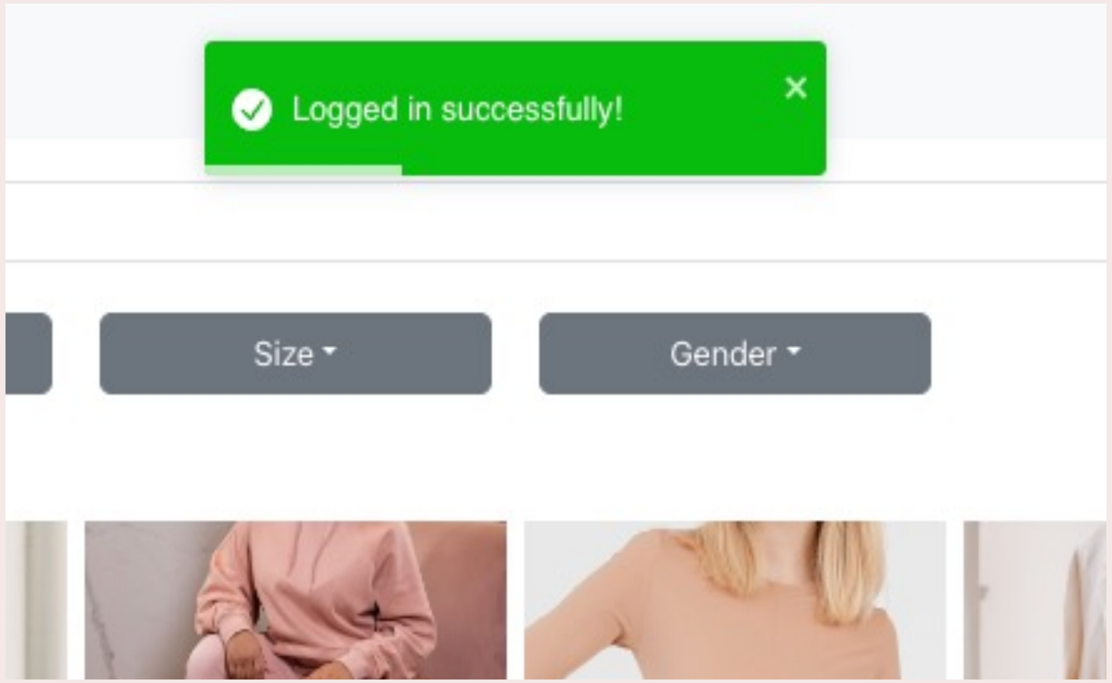
- Users are given two tokens on login
- Expired tokens do not grant access
- Expired tokens need to be refreshed by API via '/refresh'
- Intercepting requests with expired tokens allow refreshes happen seamlessly

INTERCEPTING REQUESTS & RESPONSES

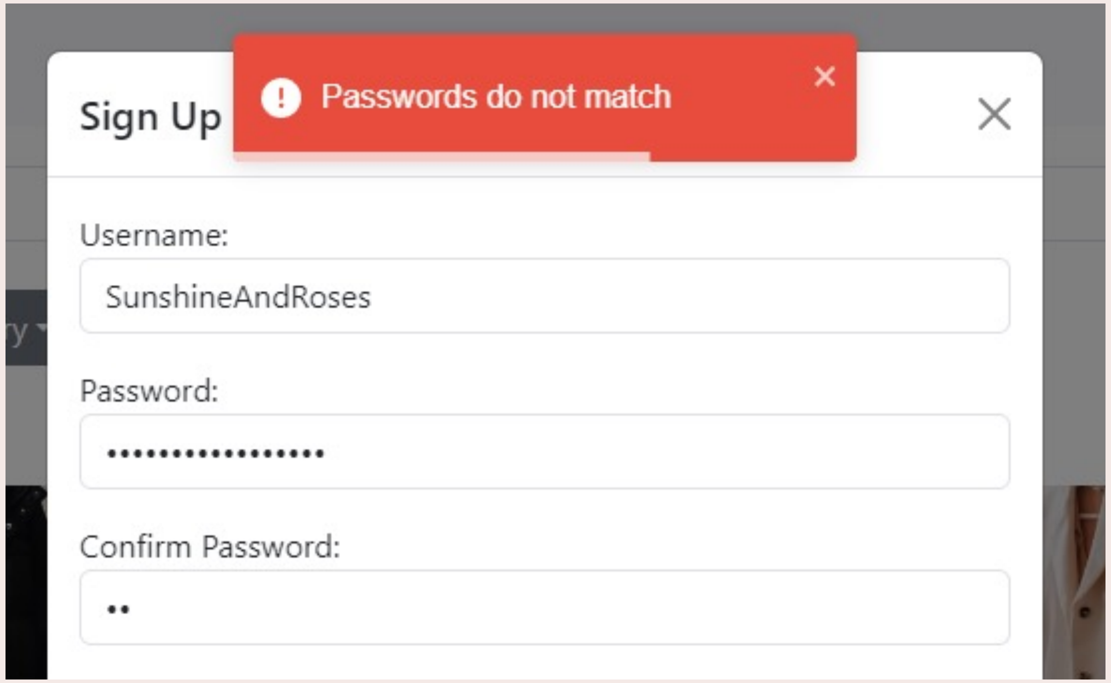


```
useEffect(() => {

    const requestIntercept = axiosJWT.interceptors.request.use(
        config => {
            if (!config.headers['authorization']) {
                config.headers['authorization'] = `Bearer ${token}`;
            }
            return config;
        }, (error) => Promise.reject(error)
    );

    const responseIntercept = axiosJWT.interceptors.response.use(
        response => response,
        async (error) => {
            const prevRequest = error?.config;
            if ((error?.response?.status === 403) && !prevRequest?.sent) {
                prevRequest.sent = true;
                const response = await axios.get('/auth/refresh',
                {
                    headers:
                        { authorization: `Bearer ${refreshToken}`}
                });


                prevRequest.headers['authorization'] = `Bearer ${response.data.token}`;
                sessionStorage.setItem('token', response.data.token);
                return axiosJWT(prevRequest);
            }
            return Promise.reject(error);
        }
    );

    return () => {
        axiosJWT.interceptors.request.eject(requestIntercept);
        axiosJWT.interceptors.response.eject(responseIntercept);
    }
})
```

SOLI: FLASH MESSAGES



Logged in successfully!



Passwords do not match

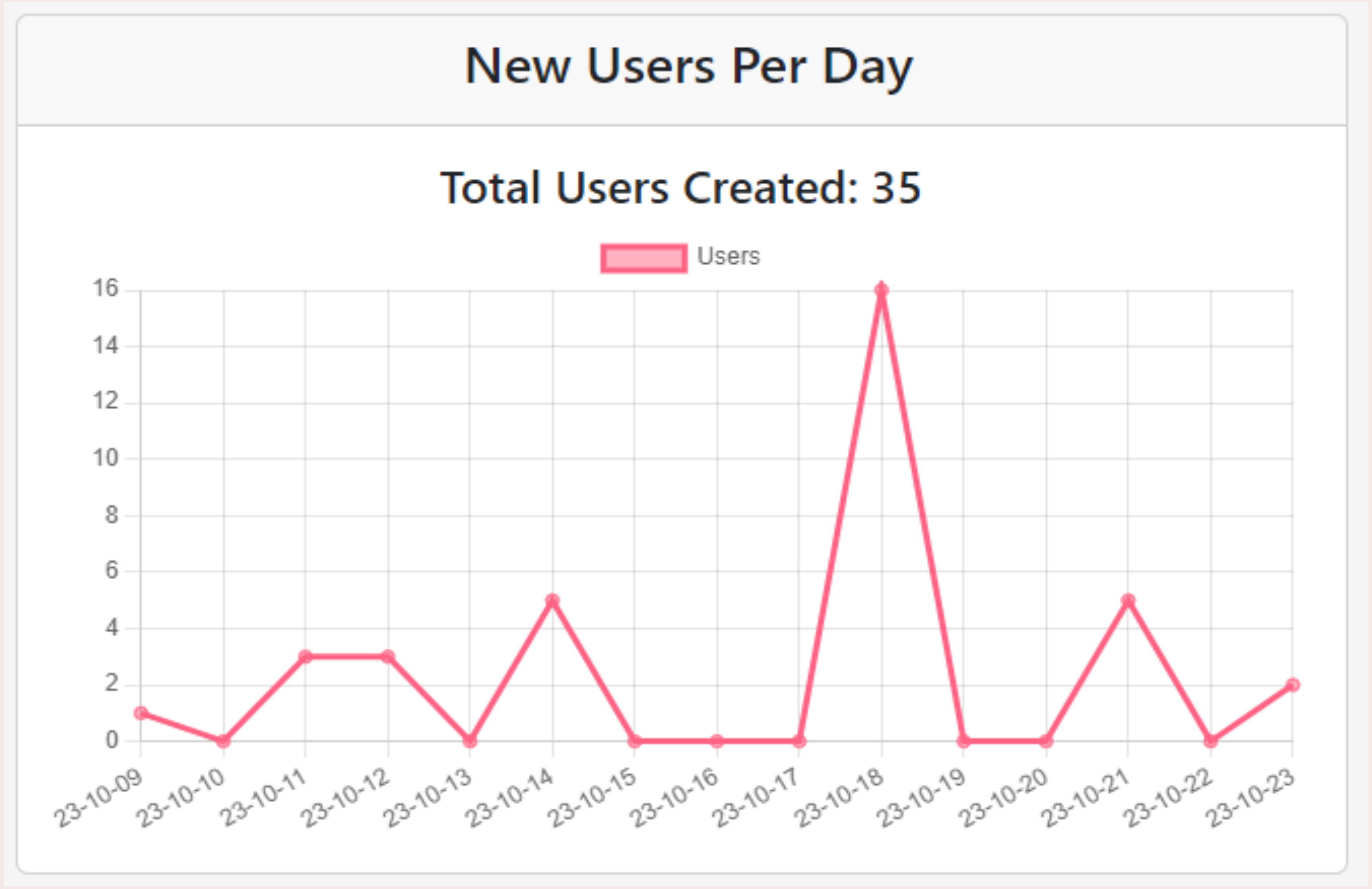
Sign Up

Username: SunshineAndRoses

Password: •••••••••••••••••

Confirm Password: ••

SOLI: CHARTS


New Users Per Day

Total Users Created: 35

FUTURE WORK

Additional features:

- Shopping carts
- Stripe payments
- Likes, follows, messages

SUMMARY

Successes:

- Simple yet cohesive
- Realistic look and feel
- Quality over quantity