Project Documentation (Atomic eCover)

1. Introduction

This document ‘s purpose:

provide a detailed overview of the design and architecture of the Atomic eCover (MERN + Python) project.

The project aims to create a web application for users himself to create eBook covers and other photos from pre-designed mockups(psd files), without any other photo editing tools.

2. System Overview

The Atomic eCover project was developed by using the following technologies:

Front-end: React.js

Back-end: Node.js, Express.js, Python

Database: MongoDB

3. System Architecture

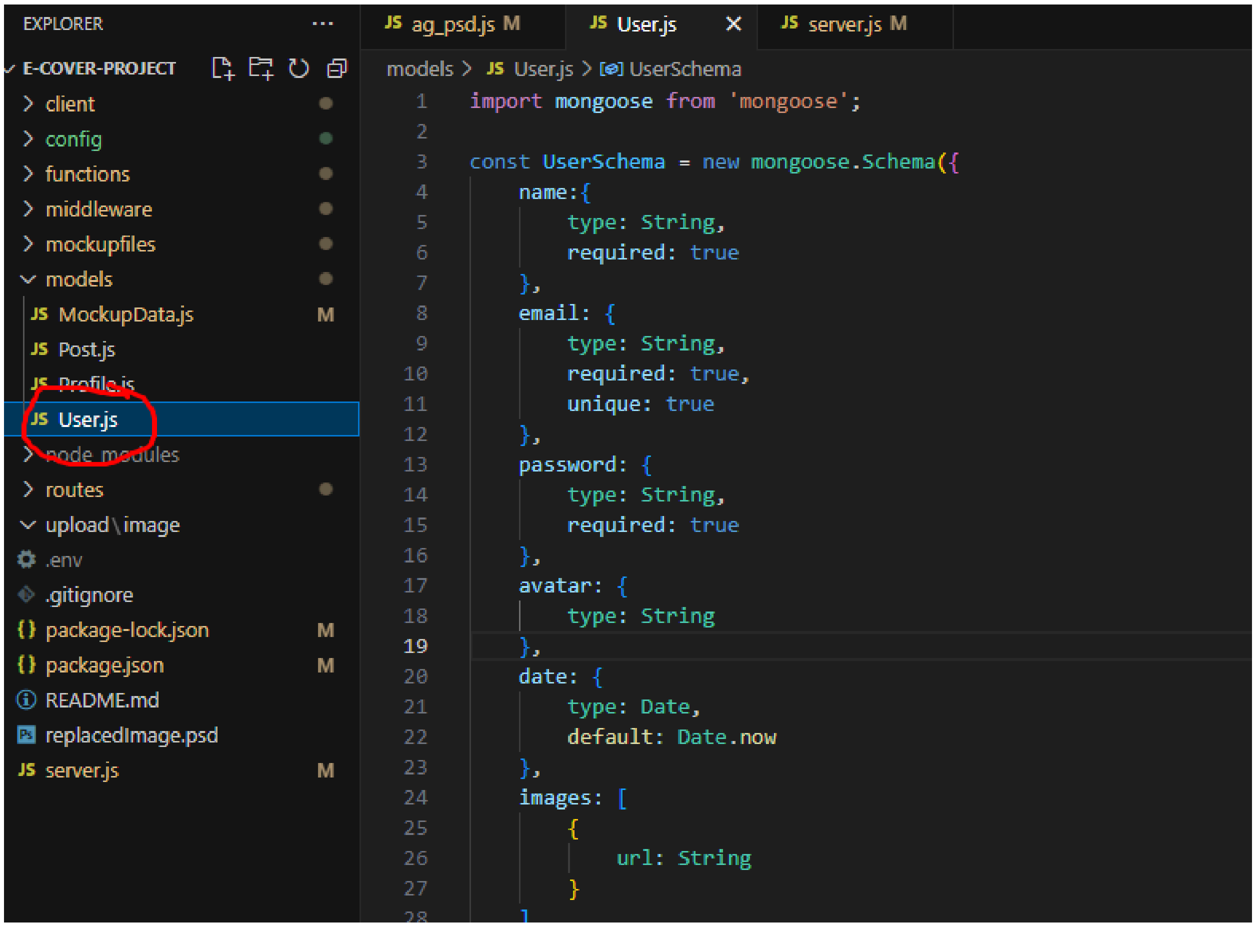
The system architecture will follow the MERN (MongoDB, Express.js, React.js, Node.js) stack. The front-end will be developed using React.js, which will communicate with the back-end API built using Node.js and Express.js. The data will be stored in a MongoDB database. Also psd-tools, python package, will be used for rendering changed psd files.

*** E-Cover design mockups comes from psd template files. But unfortunately, there is no npm or python package for controlling psd files, only exists static packages. And I couldn’t find how to change and re-render psd files. So atomic ecover project will sort out this problem by combining ag-psd npm package and psd-tools python package. ***

4. Database Design

The MongoDB database will consist of the following collections:

4.1. Users Collection


EXPLORER

E-COVER-PROJECT
- client
- config
- functions
- middleware
- mockupfiles
- models
  - MockupData.js M
  - Post.js
  - Profile.js
  - User.js
- node_modules
- routes
- upload\image
- .env
- .gitignore
- package-lock.json M
- package.json M
- README.md
- replacedImage.psd
- server.js M

ag_psd.js M | User.js | server.js M

models > User.js > UserSchema

```
import mongoose from 'mongoose';

const UserSchema = new mongoose.Schema({
    name:{
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    avatar: {
        type: String
    },
    date: {
        type: Date,
        default: Date.now
    },
    images: [
        {
            url: String
        }
    ]
```

Fields:

id: Unique identifier for the user.

name(String): Name of the user.

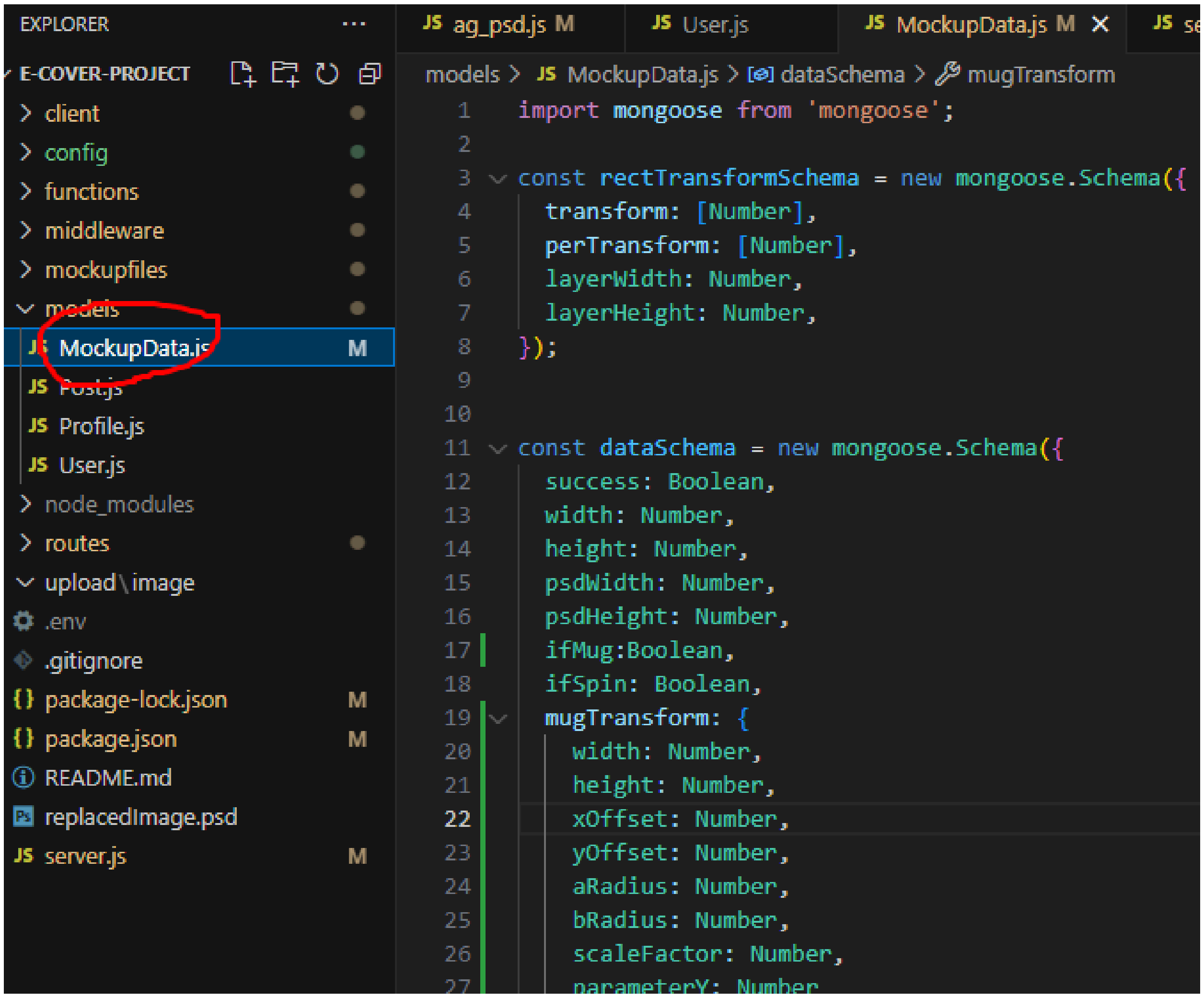
email(String): Email address of the user.

password(String): Encrypted password of the user.

images(Array): image urls uploaded by user.

covers(Array): array of saved covers infor.

4.2. MockupData Collection



```javascript
import mongoose from 'mongoose';

const rectTransformSchema = new mongoose.Schema({
  transform: [Number],
  perTransform: [Number],
  layerWidth: Number,
  layerHeight: Number,
});


const dataSchema = new mongoose.Schema({
  success: Boolean,
  width: Number,
  height: Number,
  psdWidth: Number,
  psdHeight: Number,
  ifMug:Boolean,
  ifSpin: Boolean,
  mugTransform: {
    width: Number,
    height: Number,
    xOffset: Number,
    yOffset: Number,
    aRadius: Number,
    bRadius: Number,
    scaleFactor: Number,
    parameterY: Number
```

*** This collection will be initialized when database is connected to express server.

Fields:

id: Unique identifier for the MockupData.

group(String): mockup group name.

mockups(Array): array of mockup names included to this group.

data(Array): array of mockup infors

success: Boolean,

width: Number,

height: Number,

```
psdWidth: Number,
psdHeight: Number,
ifMug:Boolean,
ifSpin: Boolean,
mugTransform: {
  width: Number,
  height: Number,
  xOffset: Number,
  yOffset: Number,
  aRadius: Number,
  bRadius: Number,
  scaleFactor: Number,
  parameterY: Number
},
spinWidth: Number,
ifSpinSplite: Boolean,
ifRectSplite: Boolean,
spinSpliteWidth: Number,
spinSpliteHeight: Number,
rectSpliteWidth: Number,
rectSpliteHeight: Number,
rectTransform: [rectTransformSchema],
spineTransform: rectTransformSchema,
spinSpliteTransform: rectTransformSchema,
rectSpliteTransform: rectTransformSchema
              rectTransformSchema
                             transform: [Number],
                             perTransform: [Number],
```

layerWidth: Number,

layerHeight: Number,

*** There are also Post and Profile schema, but those are for the future.

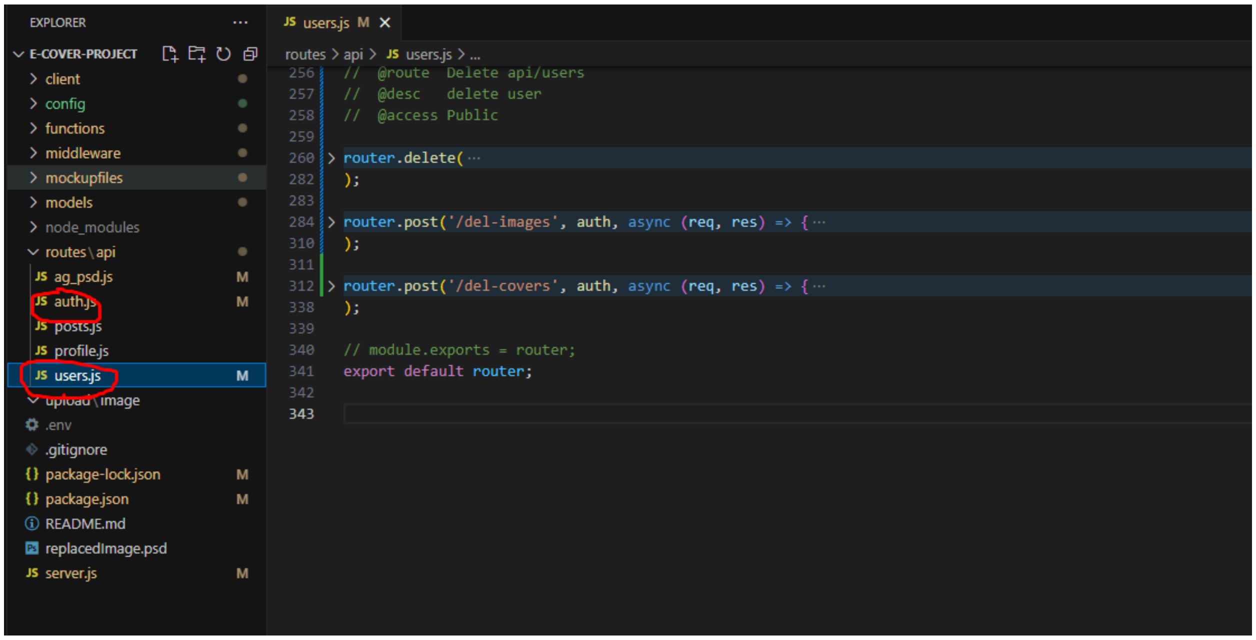
5. API Endpoints

This is an atomic eCover project ‘s API Document that describing api ‘s http method, path, request and response.

Base URL of APIs in case of dev version: http://localhost:5000/

In case of build version: domain address, ex: https://atomicecovers.com/

5.1. User Endpoints



```
//  @route  Delete api/users
//  @desc   delete user
//  @access Public

router.delete(…
);

router.post('/del-images', auth, async (req, res) => {…
);

router.post('/del-covers', auth, async (req, res) => {…
);

// module.exports = router;
export default router;
```

/ Get all users

- Method: GET

- URL: `/api/users/all-users`

- Description: Retrieve a list of all users.

- Response:

  - Content-Type: application/json
  - Body:
    - List of users (array)

/ Get user
- Method: GET
- URL: `/api/auth`
- Description: Retrieve a user.
- Response:
  - Content-Type: application/json
  - Body:
    - user

/ Register a new user
- Method: POST
- URL: `/api/users`
- Description: Create a new user.
- Request Body:
  - Content-Type: application/json
  - Parameters:
    - username (string): The username of the user.
    - email (string): The email of the user.
    - password (string): The password of the user.
- Request Body:
  - Content-Type: application/json
  - Parameters:

    - success (string): success to register.

/ Login

- Method: POST
- URL: `/api/auth`
- Description: Authenticate user and get token.
- Request Body:
  - Content-Type: application/json
  - Parameters:
    - email (string): The email of the user.
    - password (string): The password of the user.
- Response:
  - Content-Type: application/json
  - Body:
    - access_token (string): The access token for authentication.

/ Update user password

- Method: POST
- URL: `/api/users/change-pass`
- Description: Update user password.
- Request Body:
  - Content-Type: application/json
  - Parameters:
    - email (String): User email.
    - password (String): The password have to be changed.
- Response:

  - Content-Type: application/json
  - Body:
    - success (string): success to update.

/ Update user name
- Method: POST
- URL: `/api/users/change-name`
- Description: Update a user name.
- Request Body:
  - Content-Type: application/json
  - Parameters:
    - email (String): User email.
    - name (String): The name have to be changed.
- Response:
  - Content-Type: application/json
  - Body:
    - success (string): success to update.

/ Update user name and password
- Method: POST
- URL: `/api/users/change-name-pass`
- Description: Update a user name and password.
- Request Body:
  - Content-Type: application/json
  - Parameters:
    - email (String): User email.

    - name (String): The name have to be changed.
    - password (String): The password have to be changed.
- Response:
  - Content-Type: application/json
  - Body:
    - success (string): success to update.

/ Delete user
- Method: DELETE
- URL: `/api/users/:email`
- Description: Delete a user by their ID.

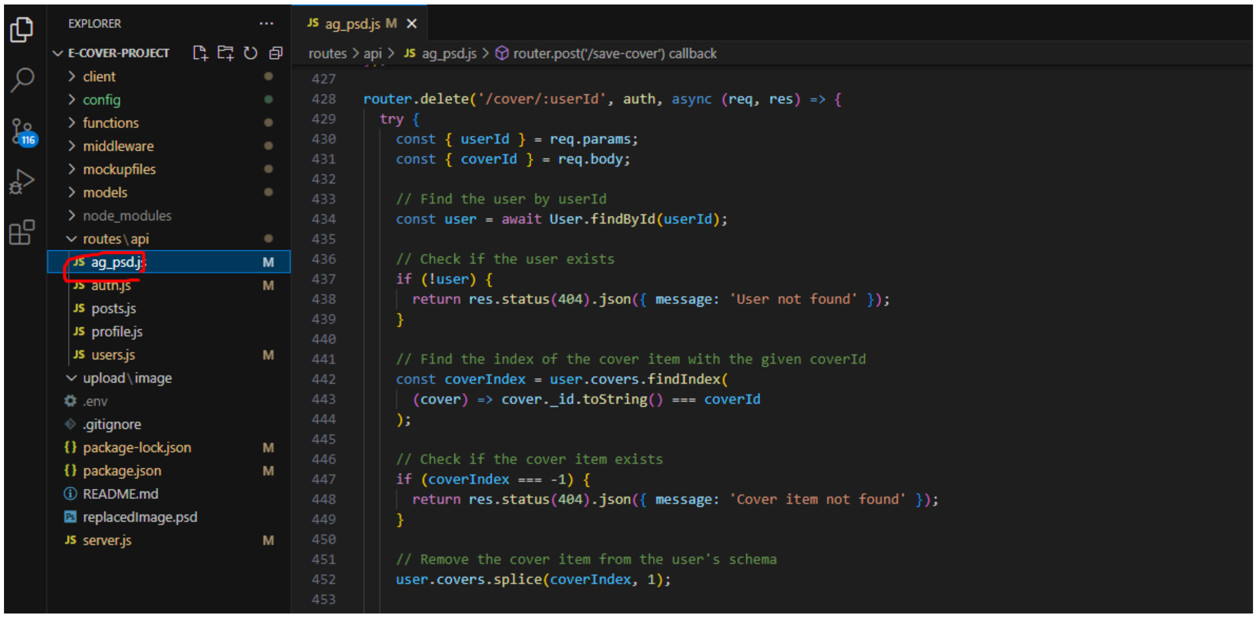
/ Delete user
- Method: DELETE
- URL: `/api/users/:email`
- Description: Delete a user by their ID.

/ Delete uploaded images
- Method: POST
- URL: `/api/users/del-images`
- Description: Delete all uploaded images.

/ Delete saved covers
- Method: POST
- URL: `/api/users/del-covers`

- Description: Delete all saved covers.

5.2. PSD and Image Feature Endpoints



```
router.delete('/cover/:userId', auth, async (req, res) => {
  try {
    const { userId } = req.params;
    const { coverId } = req.body;

    // Find the user by userId
    const user = await User.findById(userId);

    // Check if the user exists
    if (!user) {
      return res.status(404).json({ message: 'User not found' });
    }

    // Find the index of the cover item with the given coverId
    const coverIndex = user.covers.findIndex(
      (cover) => cover._id.toString() === coverId
    );

    // Check if the cover item exists
    if (coverIndex === -1) {
      return res.status(404).json({ message: 'Cover item not found' });
    }

    // Remove the cover item from the user's schema
    user.covers.splice(coverIndex, 1);

```

/ Get all Mockup datas

- Method: POST
- URL: `/api/ag-psd/all-mockup`
- Description: Retrieve a all mockup datas.
- Response:
  - Content-Type: application/json
  - Body:
    - all mockup datas

/ Get Mockup data by mockup name

- Method: GET
- URL: `/api/ag-psd/mockup/:mockup`

- Description: Retrieve a mockup data by it’s name.
- Response:
    - Content-Type: application/json
    - Body:
        - Mockup data

/ Upload Image

- Method: POST
- URL: `/api/ag-psd/upload-image`
- Description: Upload image to aws s3 bucket.
- Request Body:
    - Content-Type: application/json
    - Parameters:
        - file(FormData): image file
- Response:
    - Content-Type: application/json
    - Parameters:
        - success (string): success to register.

/ Get all uploaded images

- Method: GET
- URL: `/api/ag-psd/all-upload-image`
- Description: get all images uploaded by user.
- Response:
    - Content-Type: application/json
    - Parameters:
        - List of image urls.

/ Change and re-render Mockup

- Method: POST
- URL: `/api/ag-psd/render-image`
- Description: Replace layer images and re-render psd file.
- Request Body:
  - Content-Type: application/json
  - Parameters:
    - rectImage(Image Data): Image Data have to be changed with mm_image:YourImage layer.
    - spineImage(Image Data): Image Data have to be changed with mm_image:Spine layer.
    - rectSpliteImage(Image Data): Image Data have to be changed with mm_image:RectSplite layer.
    - spineSpliteImage(Image Data): Image Data have to be changed with mm_image:SpineSplite layer.
    - name(String): mockup name
    - ifSpine(Boolean)
- Response
  - Content-Type: application/json
  - Parameters:
    - renderedIamge(ImageData): final image data.

/ Save Cover

- Method: POST

- URL: `/api/ag-psd/save-cover`
- Description: Save current mockup design state.
- Request Body:
  - Content-Type: application/json
  - Parameters:
    - userId(ObjectId): user id
    - renderedImage(Image Data): rendered image data
    - designState(JSON data): image editor's current design state
    - mockup(JSON data): Mockup data
- Response:
  - Content-Type: application/json
  - Parameters:
    - success (string): success to save.

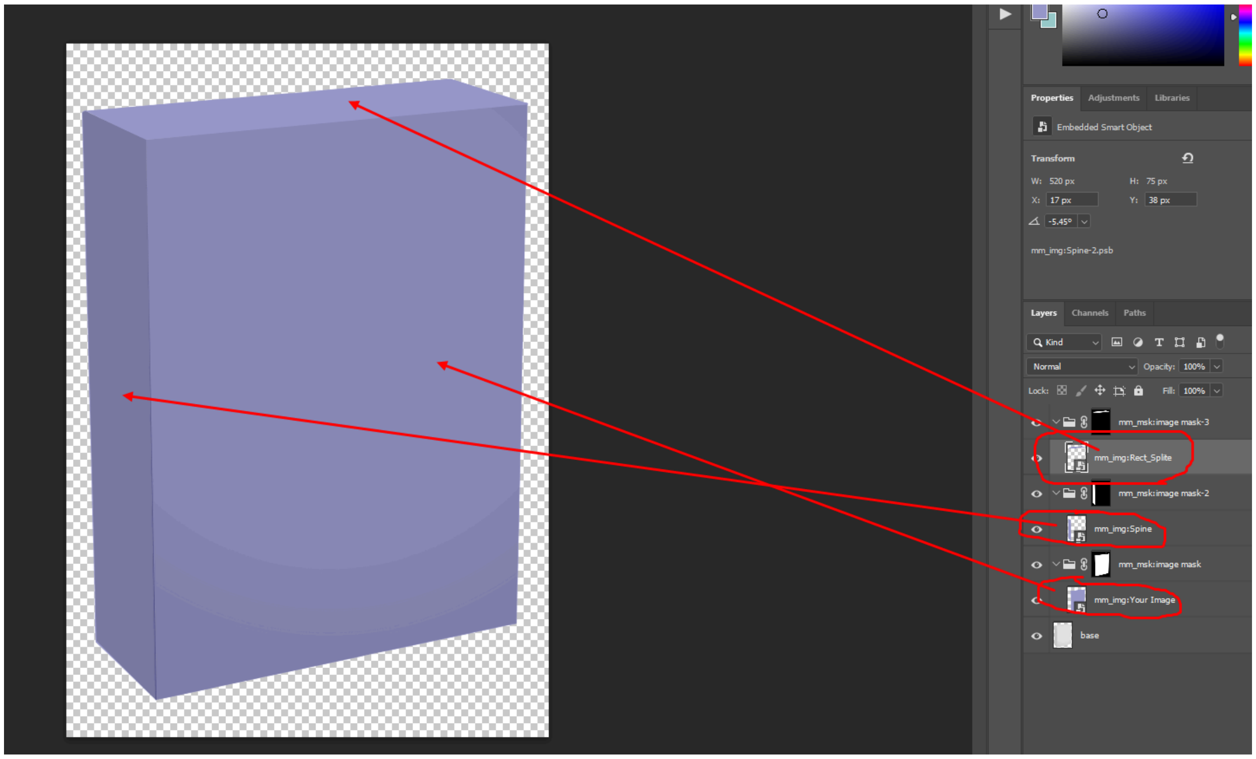
/ Get Covers

- Method: GET
- URL: `/api/ag-psd/get-covers/:userid`
- Description: Get all covers of user.
- Response:
  - Content-Type: application/json
  - Parameters:
    - List of covers data.

6. Getting Mockup Data

6.1. PSD file structure

- Layers



Embedded Smart Object

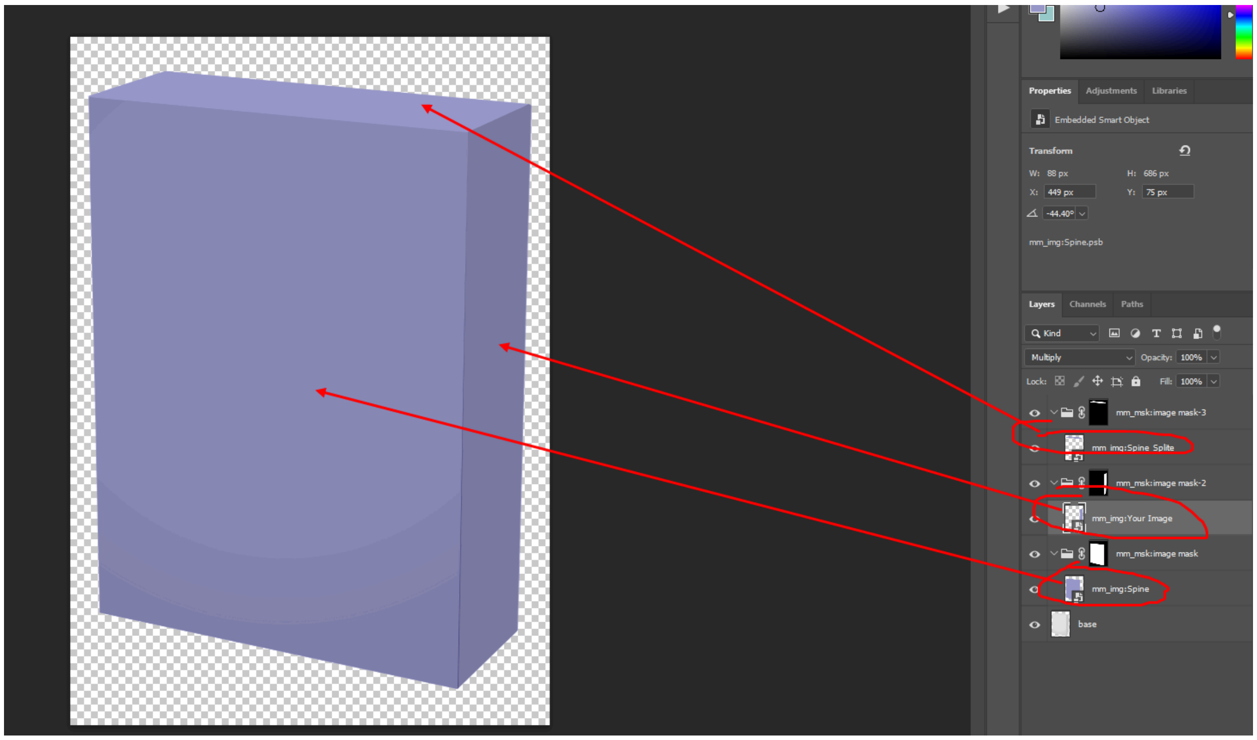
Transform

W: 520 px H: 75 px
X: 17 px Y: 38 px
-5.45°

mm_img:Spine-2.psb

Normal Opacity: 100%
Lock: Fill: 100%

mm_msk:image mask-3
mm_img:Rect_Splite
mm_msk:image mask-2
mm_img:Spine
mm_msk:image mask
mm_img:Your Image
base



Embedded Smart Object

Transform

W: 88 px H: 686 px
X: 449 px Y: 75 px
-44.40°

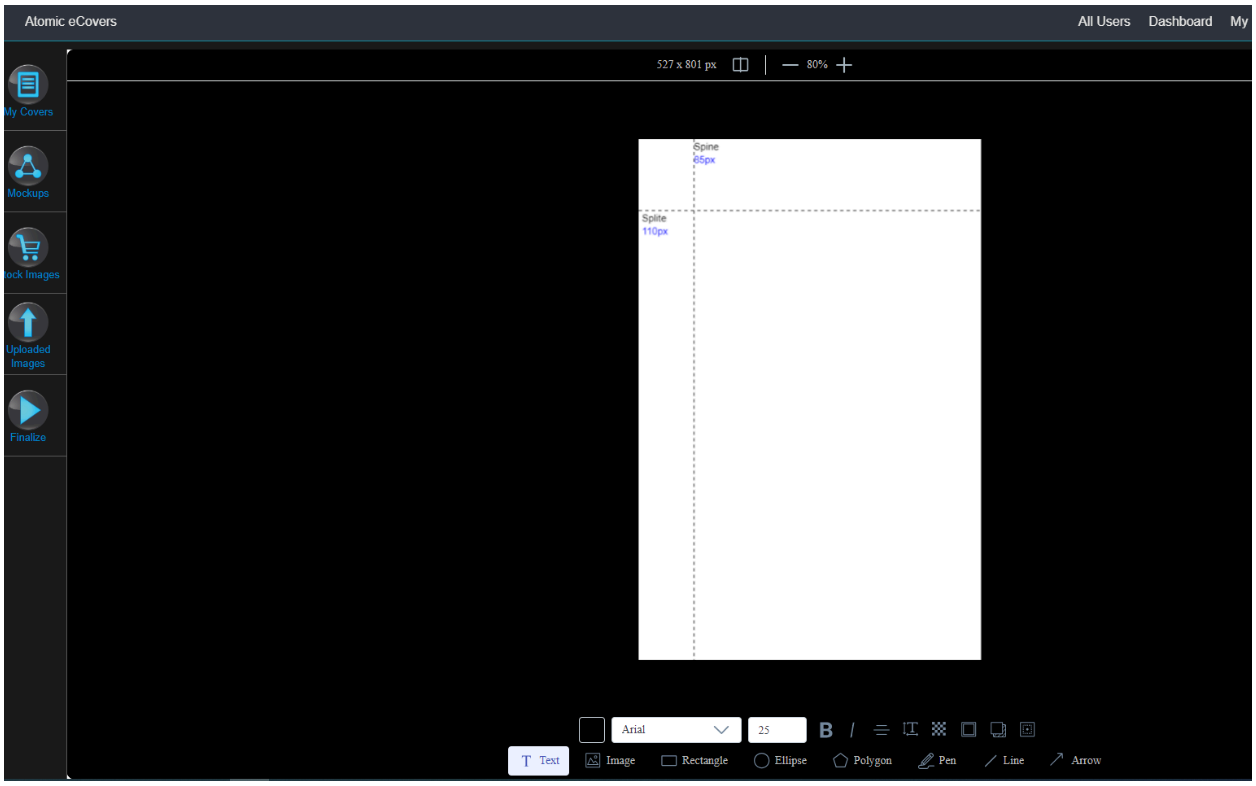
mm_img:Spine.psb

Multiply Opacity: 100%
Lock: Fill: 100%

mm_msk:image mask-3
mm_img:Spine_Splite
mm_msk:image mask-2
mm_img:Your Image
mm_msk:image mask
mm_img:Spine
base

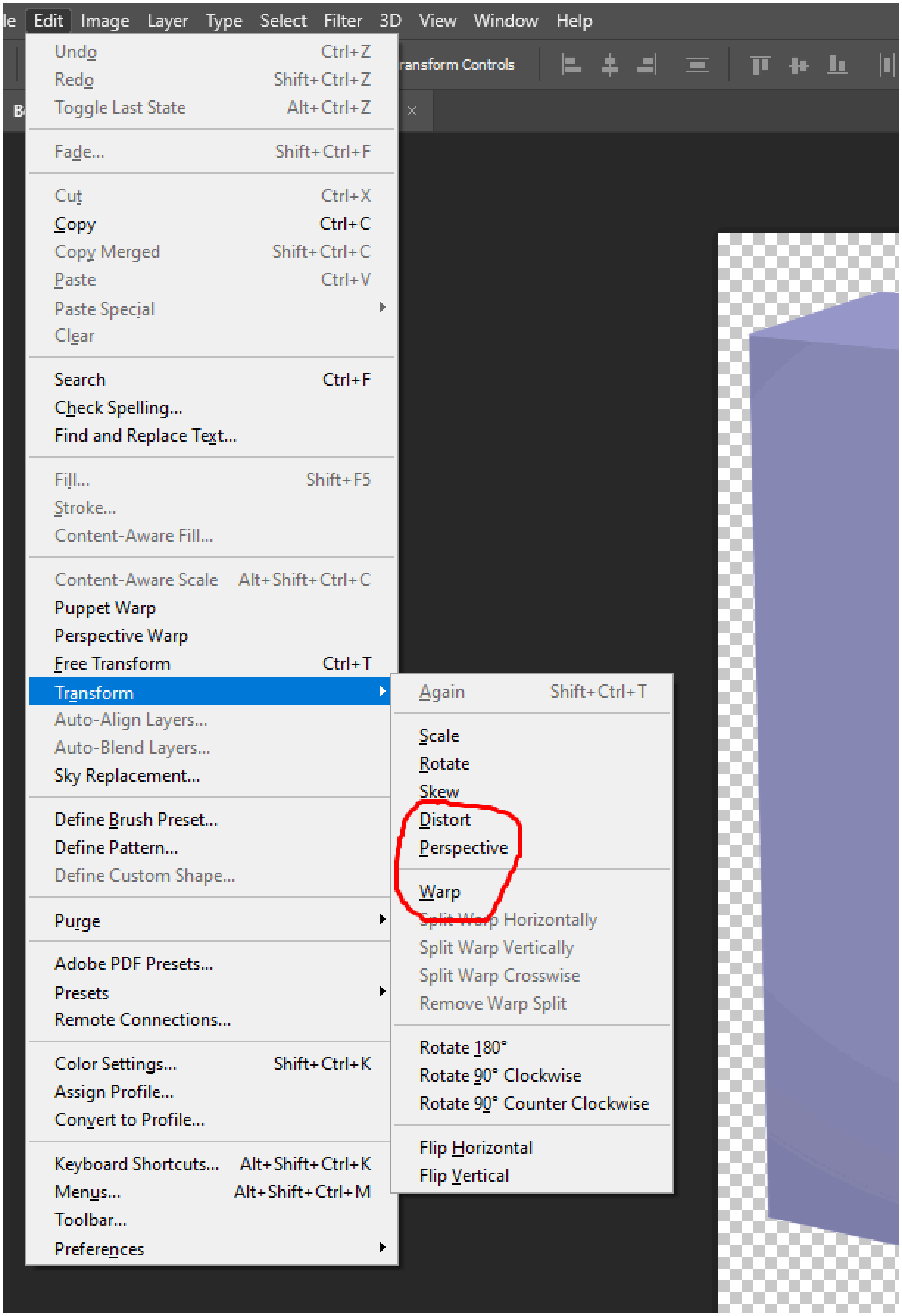
Edited Image

Have to be changed with mm_img:Spine_Splite layer’s one	Have to be changed with mm_img:Rect_Splite layer’s one
Have to be changed with mm_img:Spine layer’s one	Have to be changed with mm_img:Your Image layer’s one

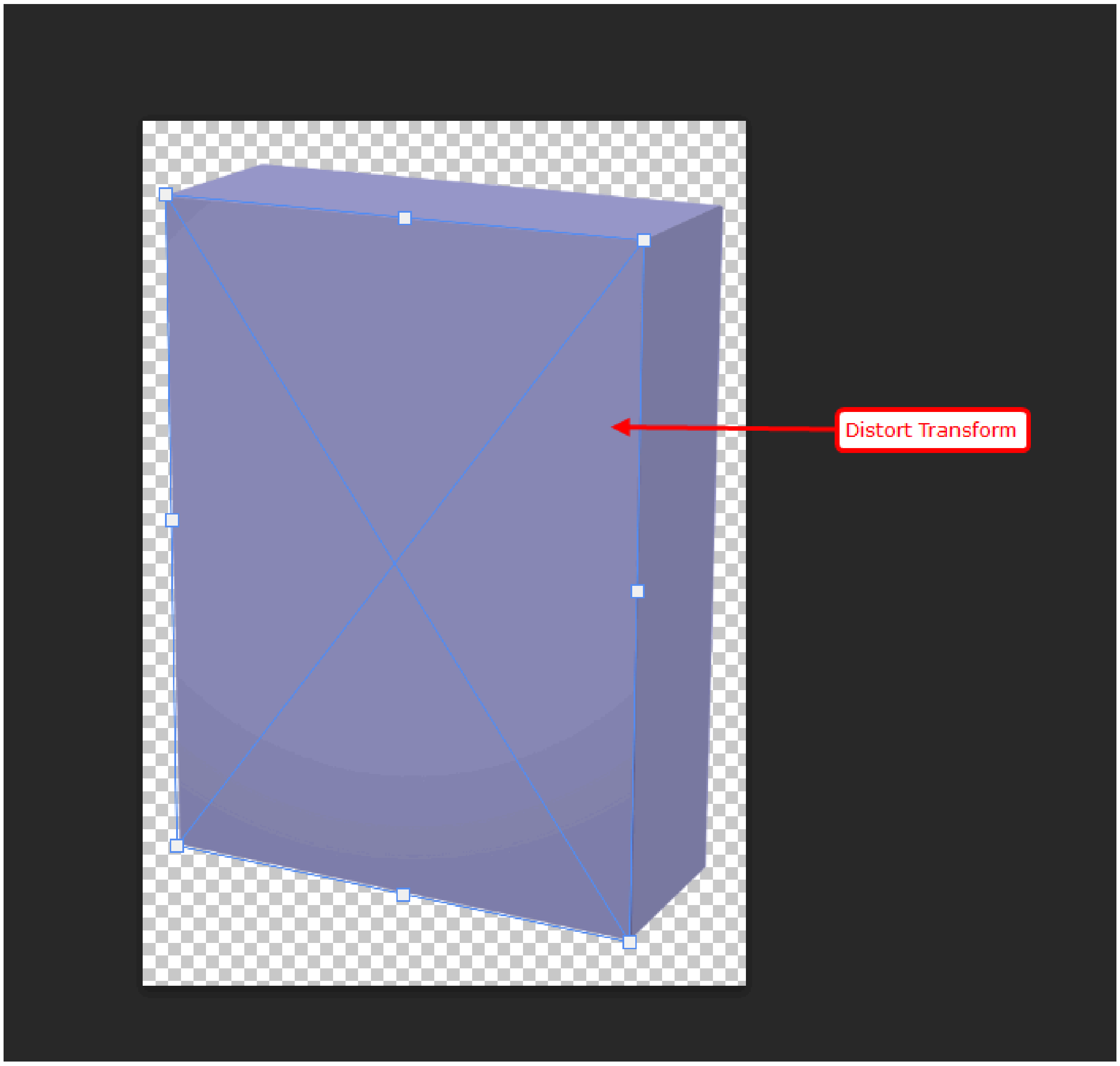


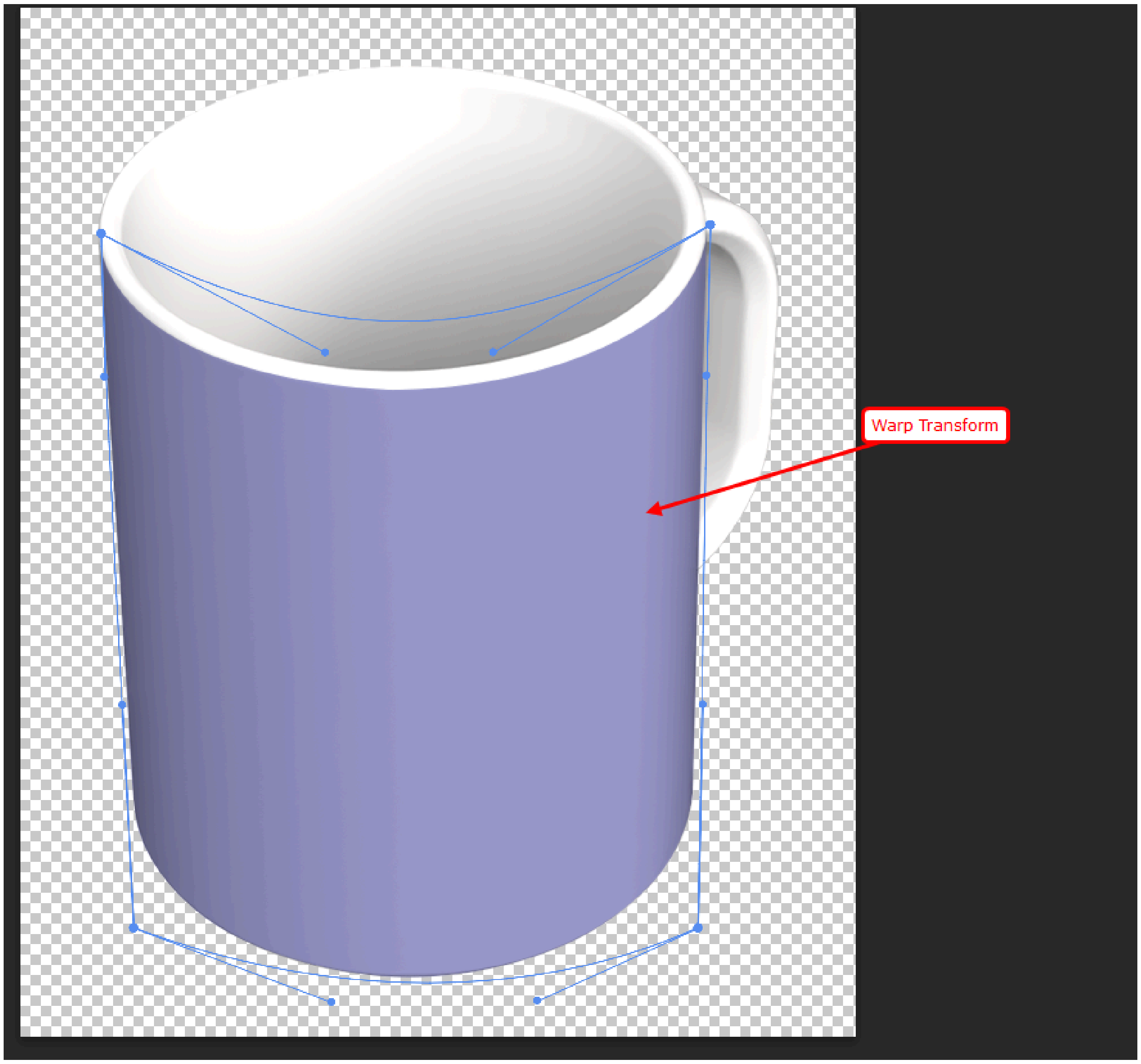
- Transform

There are several transforms in layers, including perspective, distort and warp.

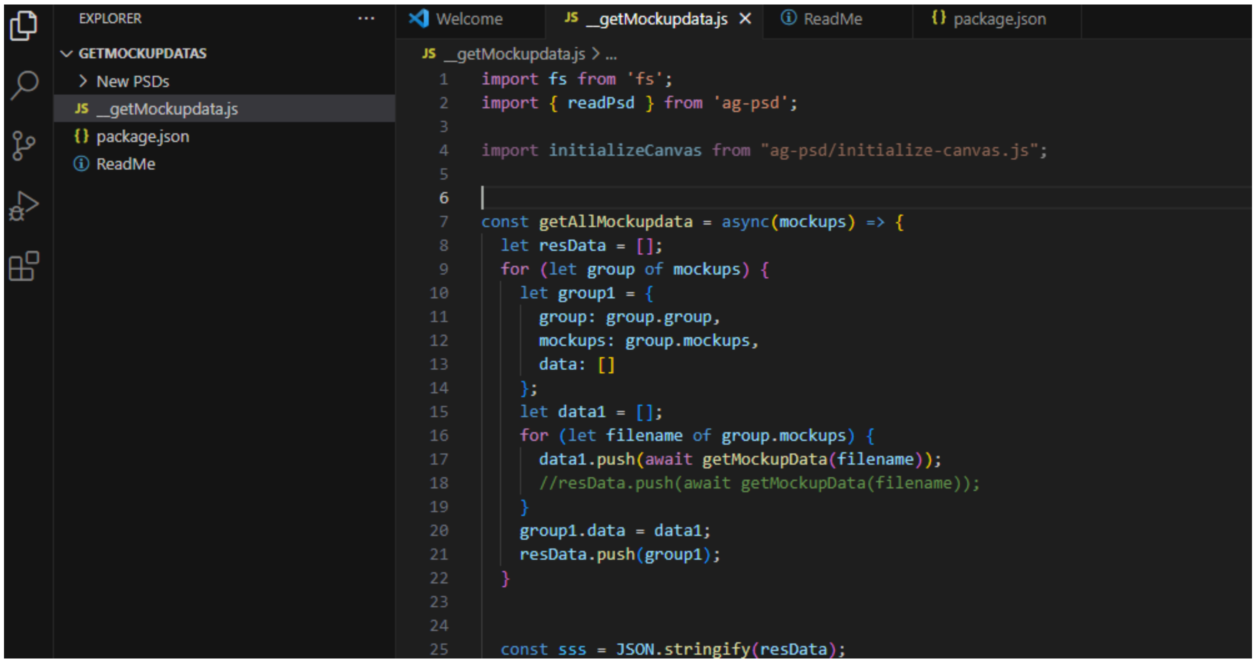


Edit
Image
Layer
Type
Select
Filter
3D
View
Window
Help
ransform Controls
Undo Ctrl+Z
Redo Shift+Ctrl+Z
Toggle Last State Alt+Ctrl+Z
Fade... Shift+Ctrl+F
Cut Ctrl+X
Copy Ctrl+C
Copy Merged Shift+Ctrl+C
Paste Ctrl+V
Paste Special
Clear
Search Ctrl+F
Check Spelling...
Find and Replace Text...
Fill... Shift+F5
Stroke...
Content-Aware Fill...
Content-Aware Scale Alt+Shift+Ctrl+C
Puppet Warp
Perspective Warp
Free Transform Ctrl+T
Transform
Auto-Align Layers...
Auto-Blend Layers...
Sky Replacement...
Define Brush Preset...
Define Pattern...
Define Custom Shape...
Purge
Adobe PDF Presets...
Presets
Remote Connections...
Color Settings... Shift+Ctrl+K
Assign Profile...
Convert to Profile...
Keyboard Shortcuts... Alt+Shift+Ctrl+K
Menus... Alt+Shift+Ctrl+M
Toolbar...
Preferences
Again Shift+Ctrl+T
Scale
Rotate
Skew
Distort
Perspective
Warp
Split Warp Horizontally
Split Warp Vertically
Split Warp Crosswise
Remove Warp Split
Rotate 180°
Rotate 90° Clockwise
Rotate 90° Counter Clockwise
Flip Horizontal
Flip Vertical


Distort Transform



6.2. Getting mockup data from psd files



```javascript
import fs from 'fs';
import { readPsd } from 'ag-psd';

import initializeCanvas from "ag-psd/initialize-canvas.js";


const getAllMockupdata = async(mockups) => {
  let resData = [];
  for (let group of mockups) {
    let group1 = {
      group: group.group,
      mockups: group.mockups,
      data: []
    };
    let data1 = [];
    for (let filename of group.mockups) {
      data1.push(await getMockupData(filename));
      //resData.push(await getMockupData(filename));
    }
    group1.data = data1;
    resData.push(group1);
  }


  const sss = JSON.stringify(resData);
```

For parsing psd mockup files and getting layers and transform data, __getMockupdata.js file was created.

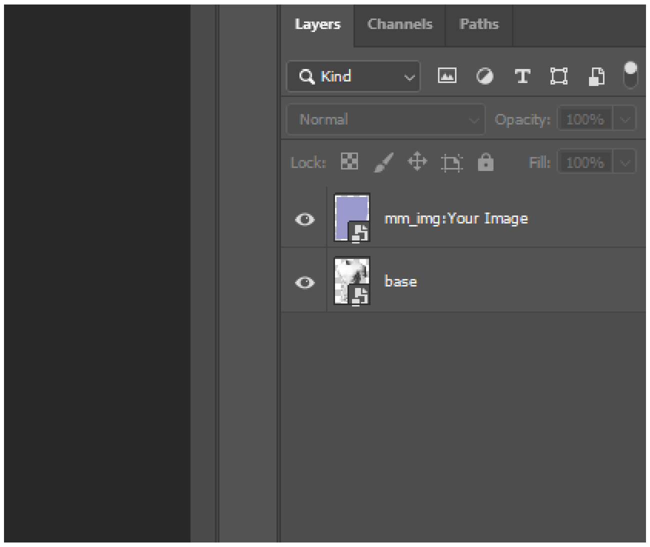
7. Adding mockups and Stock Images

7.1. Adding Mockup Templates

7.1.1 Preparing mockup psd file

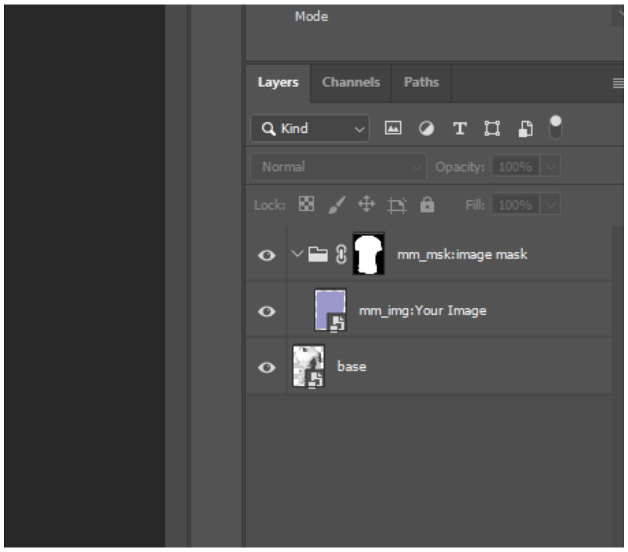
Psd file should have structure described in 6.1. Also changing layers(mm_img:Your Image, mm_img:Spine, mm_img:Spine_Splite, mm_img:Rect_Splite) should be placed at first or second level in hierarchy.

- Correct Structure



mm_img:Your Image

base

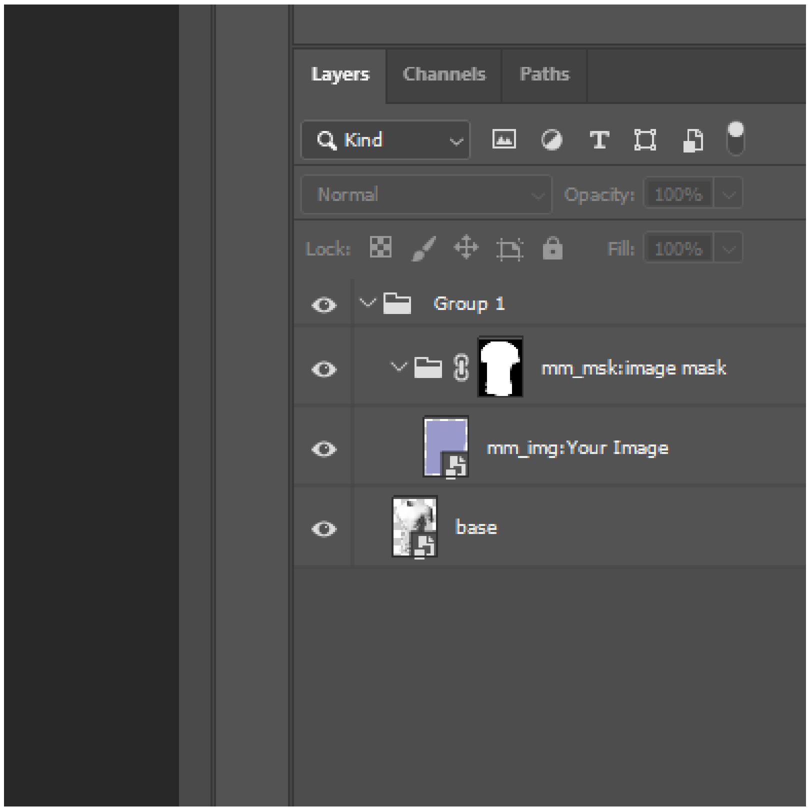


mm_msk:image mask

mm_img:Your Image

base

- Incorrect Structure



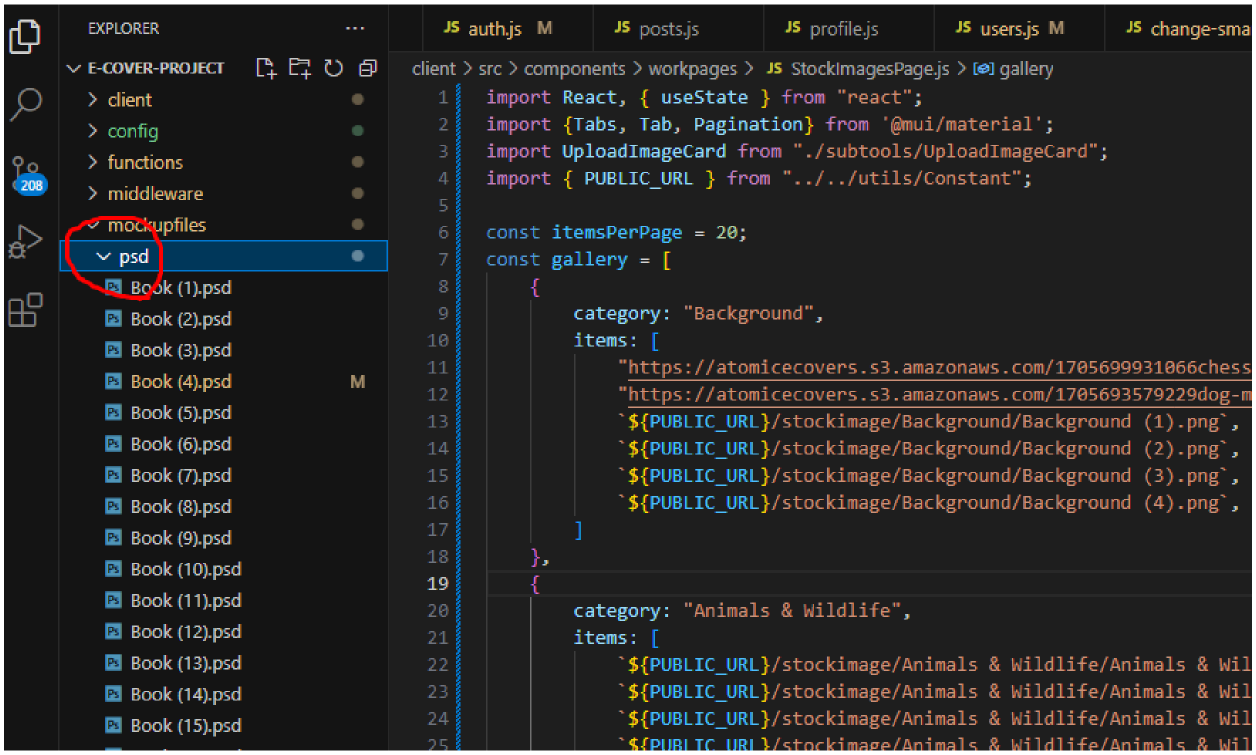
7.1.2 Adding psd and png file

- Adding psd file

Path: mockupfiles/psd/

Name: [Mockup category name] [number].psd

ex: Book (1).psd

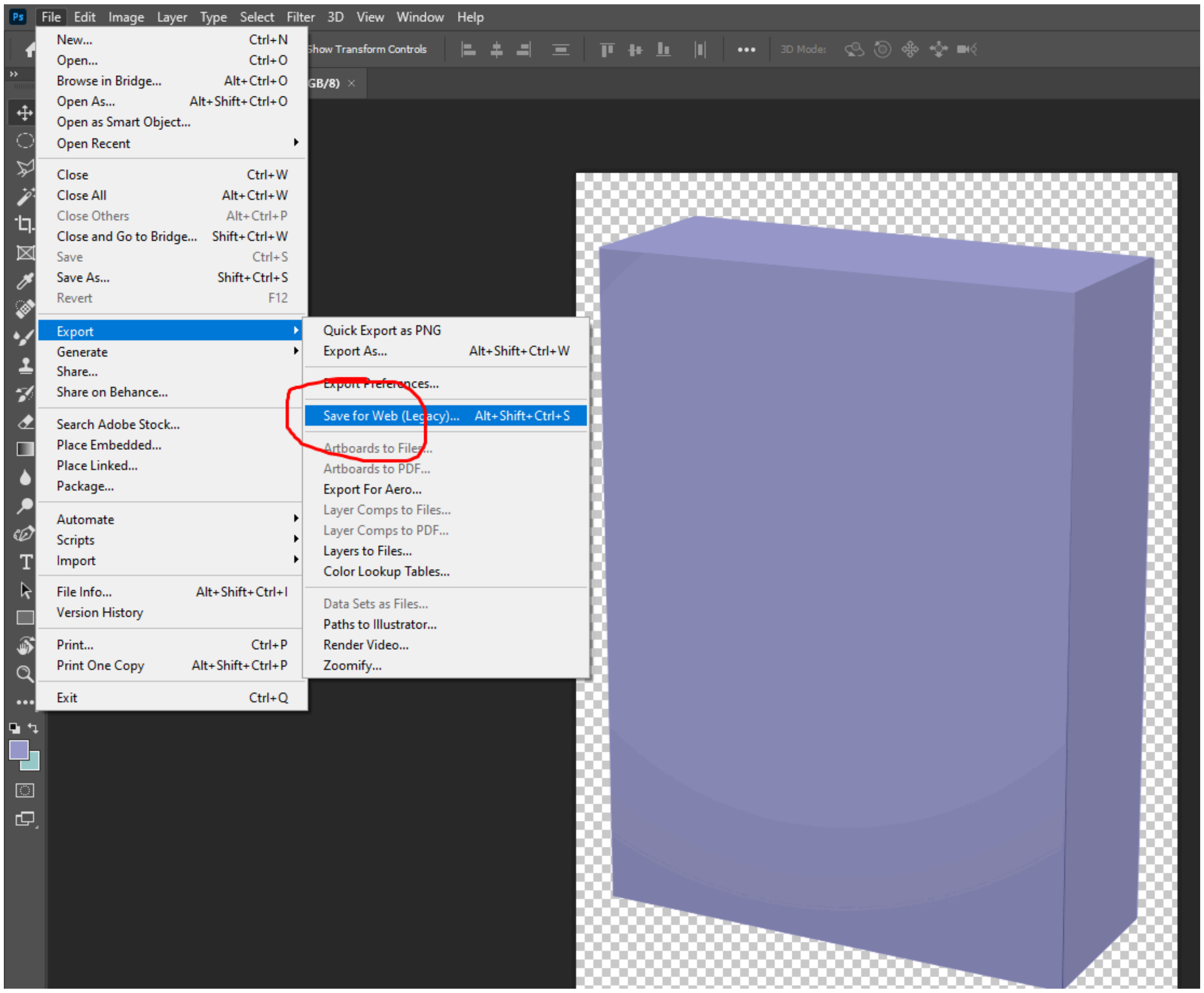


```
import React, { useState } from "react";
import {Tabs, Tab, Pagination} from '@mui/material';
import UploadImageCard from "./subtools/UploadImageCard";
import { PUBLIC_URL } from "../../utils/Constant";

const itemsPerPage = 20;
const gallery = [
    {
        category: "Background",
        items: [
            "https://atomicecovers.s3.amazonaws.com/1705699931066chess
            "https://atomicecovers.s3.amazonaws.com/1705693579229dog-m
            `${PUBLIC_URL}/stockimage/Background/Background (1).png`,
            `${PUBLIC_URL}/stockimage/Background/Background (2).png`,
            `${PUBLIC_URL}/stockimage/Background/Background (3).png`,
            `${PUBLIC_URL}/stockimage/Background/Background (4).png`,
        ]
    },
    {
        category: "Animals & Wildlife",
        items: [
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wil
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wil
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wil
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wil
```

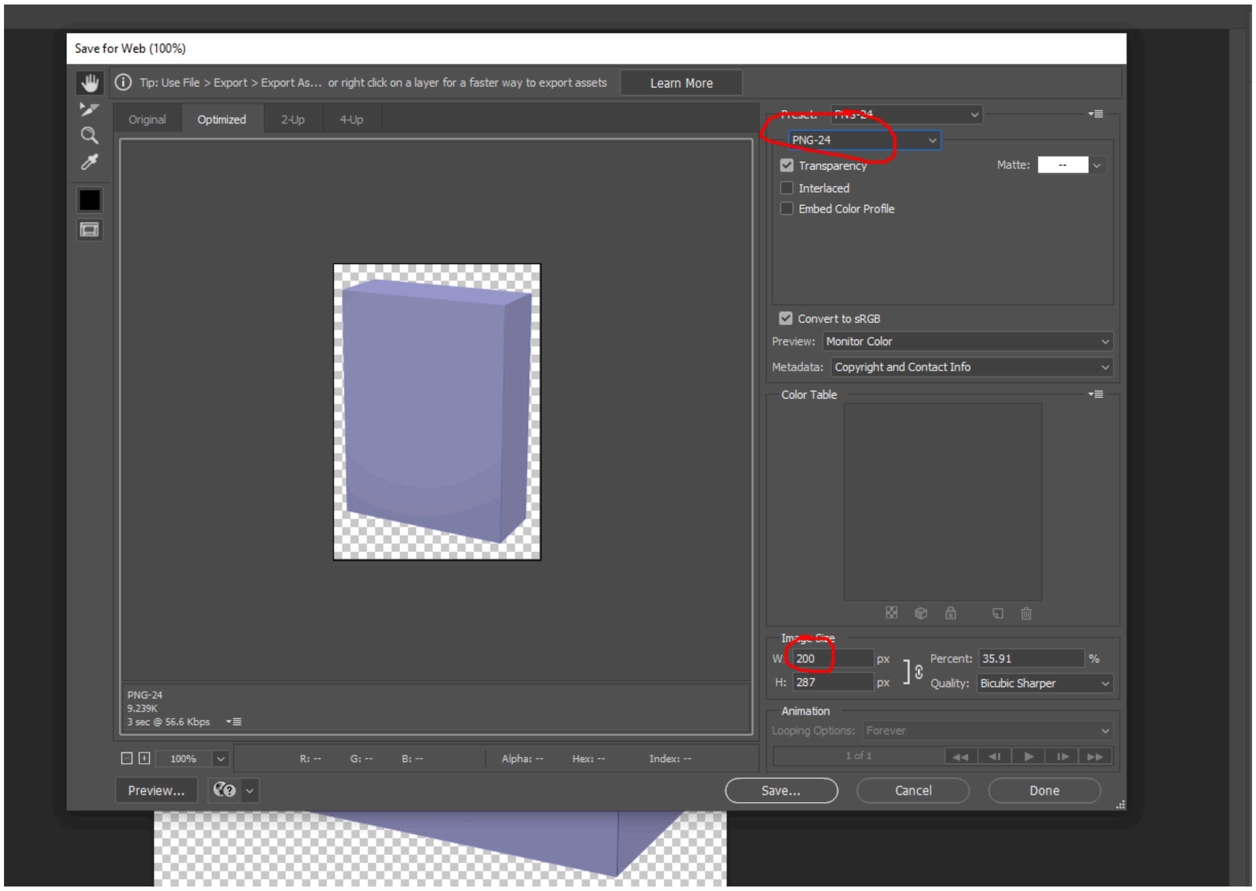
*** mockupfiles/psd folder and New PSDs folder should be same. ***

- Adding png file

Getting png file from psd file


File
Edit
Image
Layer
Type
Select
Filter
3D
View
Window
Help
New... Ctrl+N
Open... Ctrl+O
Browse in Bridge... Alt+Ctrl+O
Open As... Alt+Shift+Ctrl+O
Open as Smart Object...
Open Recent
Close Ctrl+W
Close All Alt+Ctrl+W
Close Others Alt+Ctrl+P
Close and Go to Bridge... Shift+Ctrl+W
Save Ctrl+S
Save As... Shift+Ctrl+S
Revert F12
Export
Generate
Share...
Share on Behance...
Search Adobe Stock...
Place Embedded...
Place Linked...
Package...
Automate
Scripts
Import
File Info... Alt+Shift+Ctrl+I
Version History
Print... Ctrl+P
Print One Copy Alt+Shift+Ctrl+P
Exit Ctrl+Q
Quick Export as PNG
Export As... Alt+Shift+Ctrl+W
Export Preferences...
Save for Web (Legacy)... Alt+Shift+Ctrl+S
Artboards to Files...
Artboards to PDF...
Export For Aero...
Layer Comps to Files...
Layer Comps to PDF...
Layers to Files...
Color Lookup Tables...
Data Sets as Files...
Paths to Illustrator...
Render Video...
Zoomify...
Show Transform Controls
3D Mode:

Width: 200px



Preset: PNG-24

PNG-24

[✓] Transparency

[ ] Interlaced

[ ] Embed Color Profile

Matte: --

[✓] Convert to sRGB

Preview: Monitor Color

Metadata: Copyright and Contact Info

W: 200 px

H: 287 px

Percent: 35.91 %

Quality: Bicubic Sharper

Looping Options: Forever

1 of 1

PNG-24
9.239K
3 sec @ 56.6 Kbps

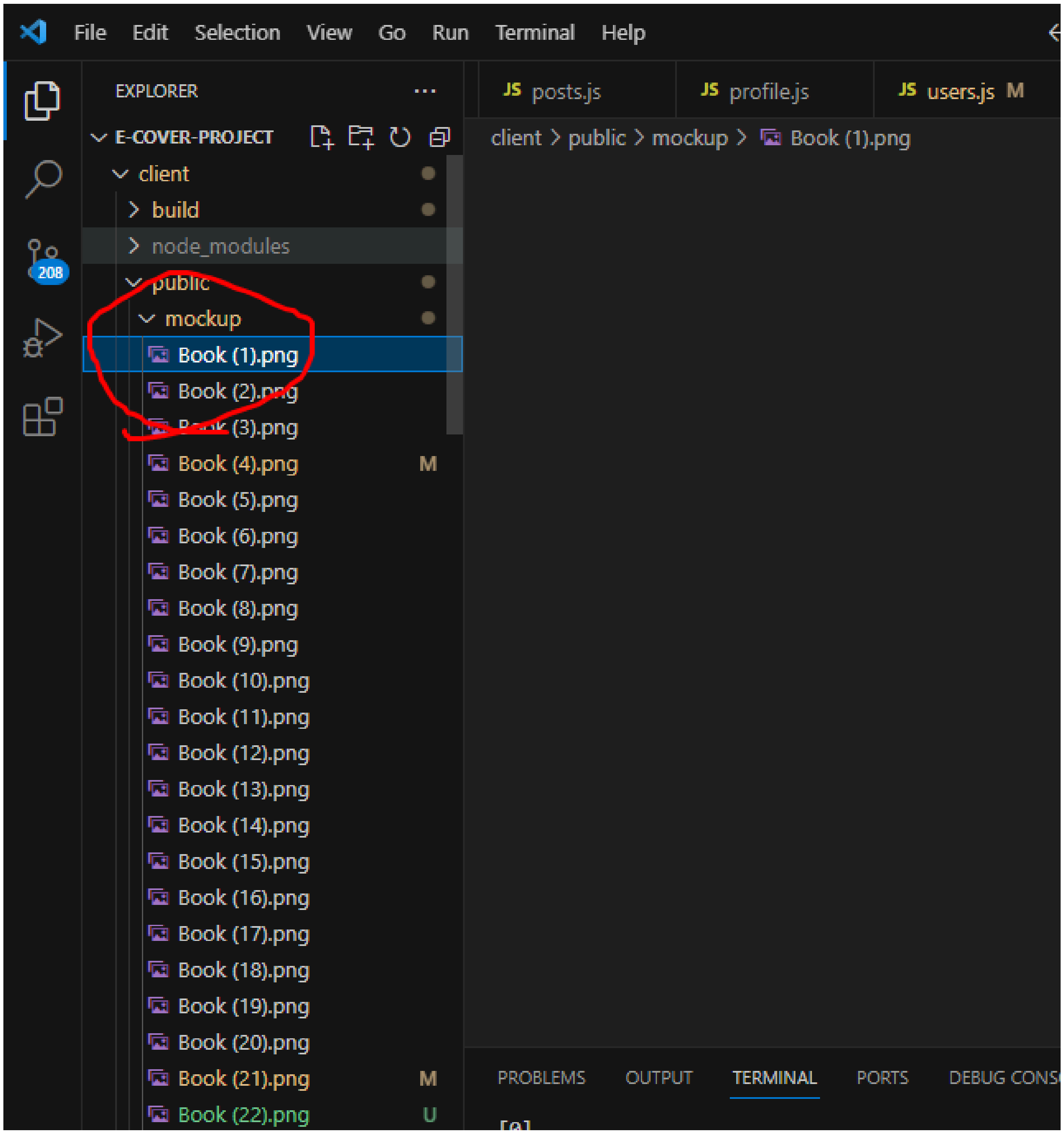
100%

R: -- G: -- B: -- Alpha: -- Hex: -- Index: --

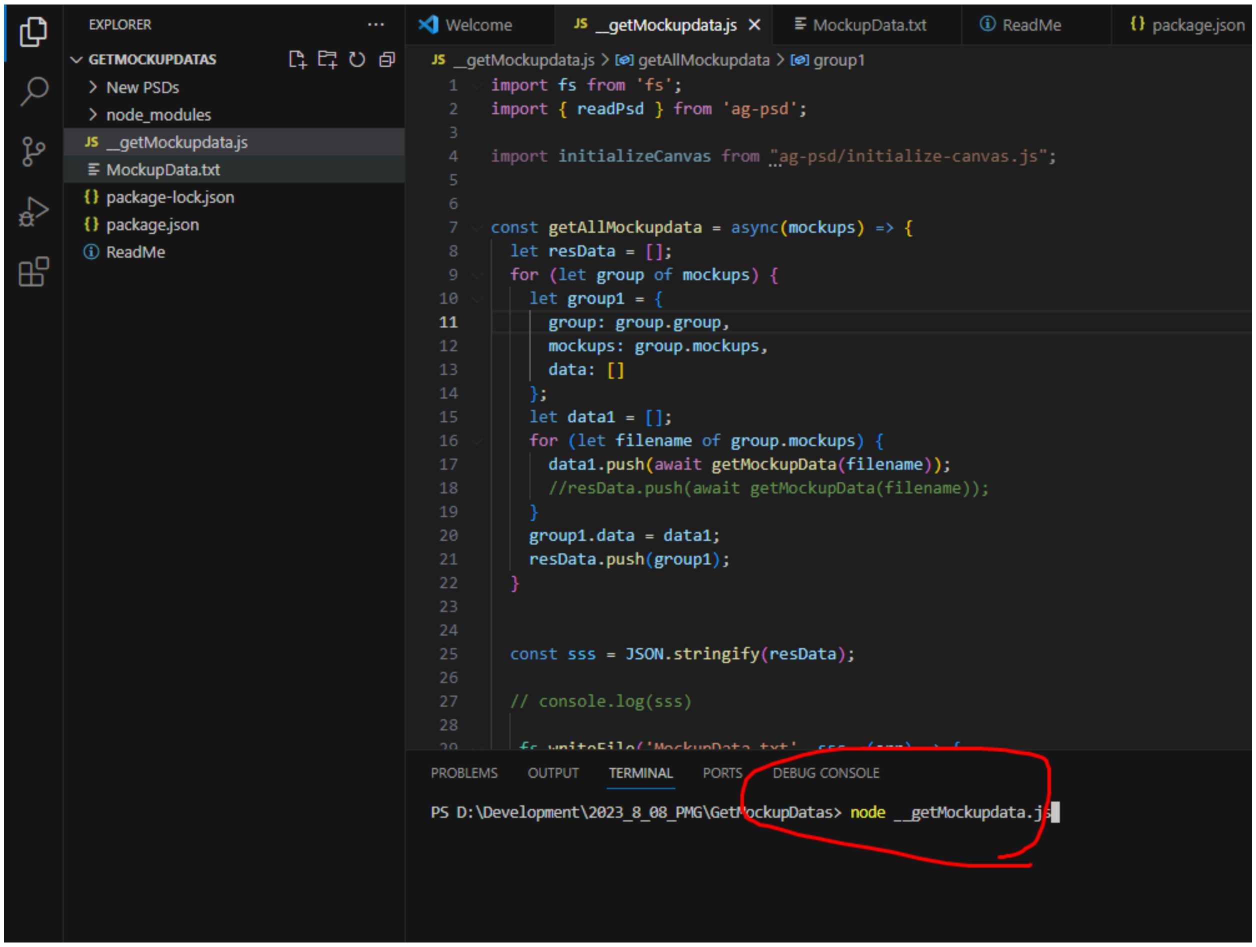
Path: client\public\mockup\

Name: [Mockup category name] [number].png

ex: Book (1).png



7.1.3 Adding mockup data



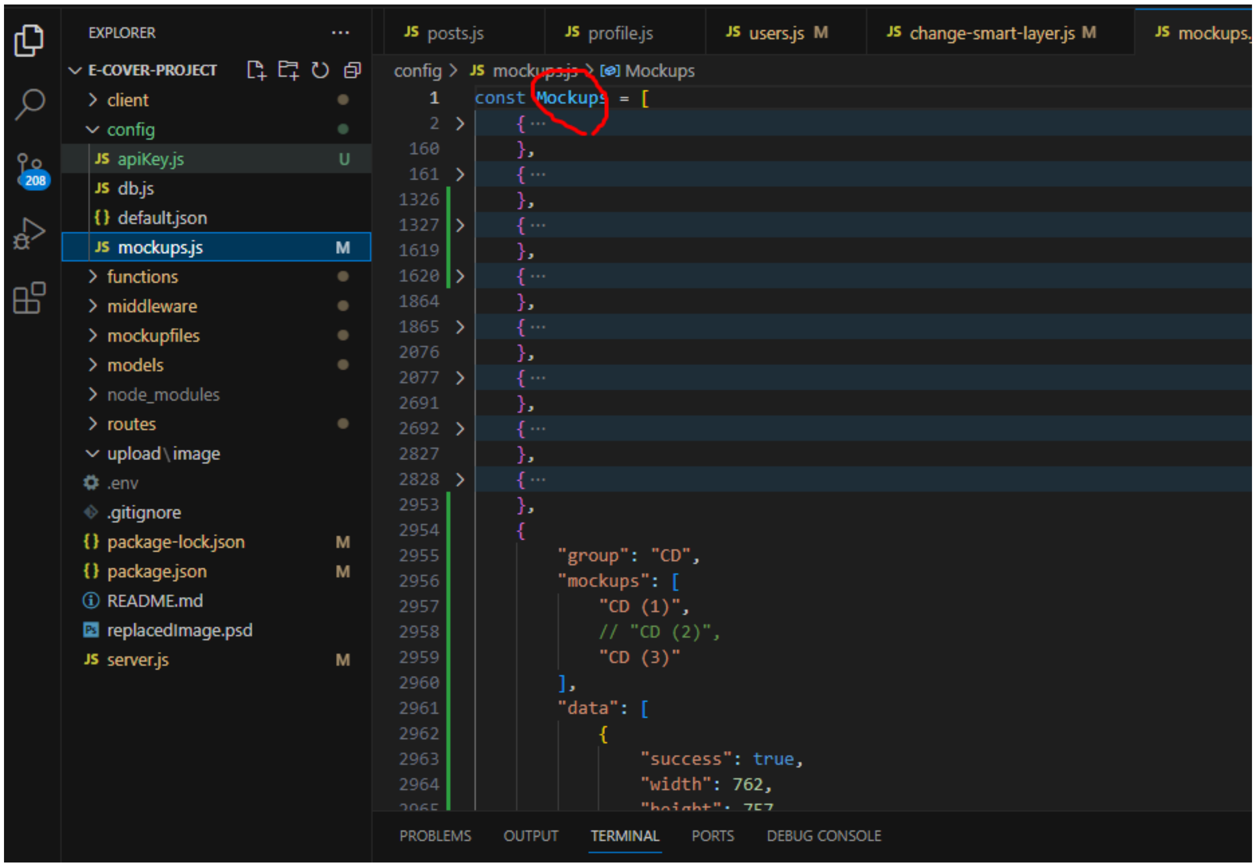
```javascript
import fs from 'fs';
import { readPsd } from 'ag-psd';

import initializeCanvas from "ag-psd/initialize-canvas.js";


const getAllMockupdata = async(mockups) => {
  let resData = [];
  for (let group of mockups) {
    let group1 = {
      group: group.group,
      mockups: group.mockups,
      data: []
    };
    let data1 = [];
    for (let filename of group.mockups) {
      data1.push(await getMockupData(filename));
      //resData.push(await getMockupData(filename));
    }
    group1.data = data1;
    resData.push(group1);
  }


  const sss = JSON.stringify(resData);

  // console.log(sss)

```

```
PS D:\Development\2023_8_08_PMG\GetMockupDatas> node __getMockupdata.js
```

Then,

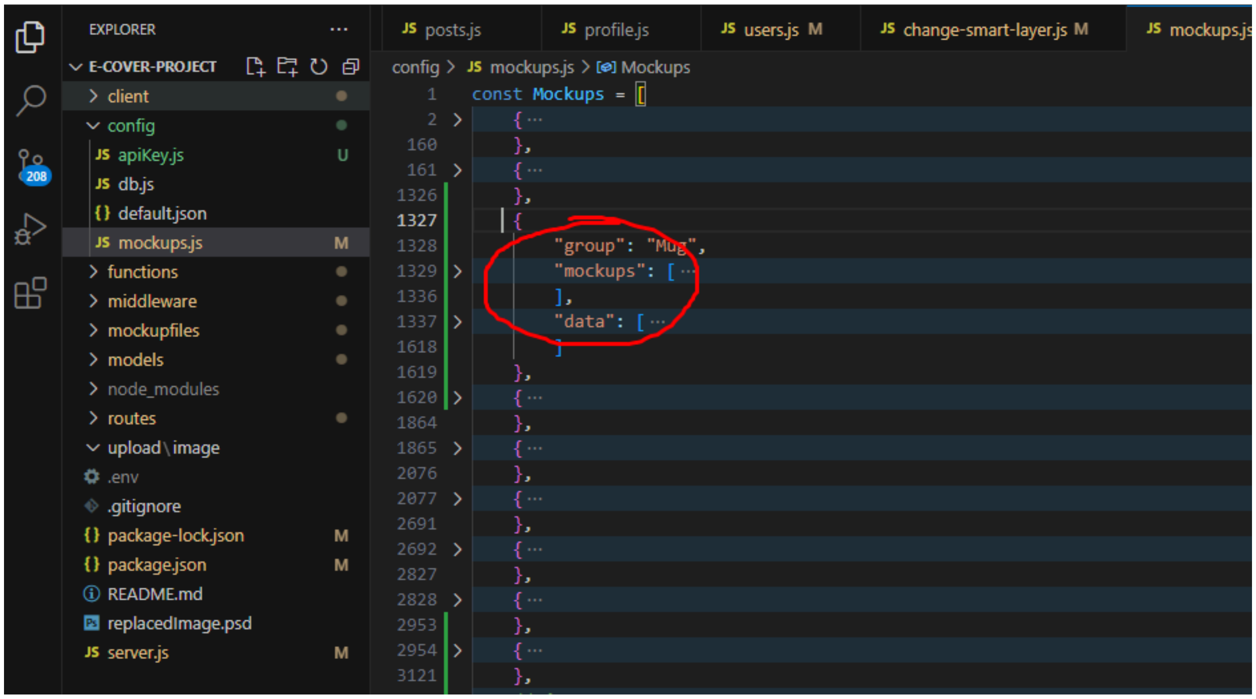
Mockup datas will be stored in MockupData.txt.

Just copy this json data and paste as const variable Mockups in config/mockup.js file.

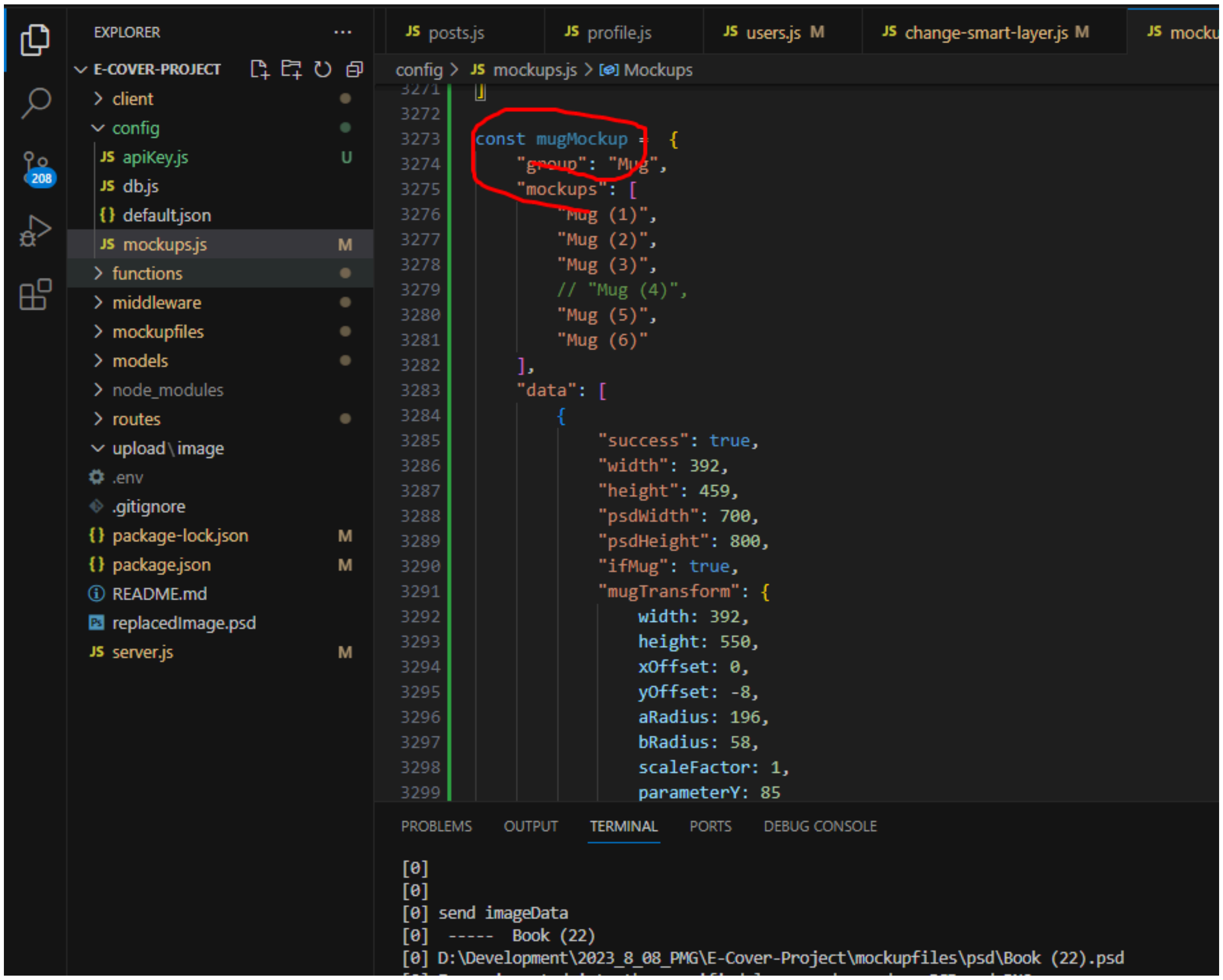


```
const Mockups = [
    {…
    },
    {…
    },
    {…
    },
    {…
    },
    {…
    },
    {…
    },
    {…
    },
    {…
    },
    {
        "group": "CD",
        "mockups": [
            "CD (1)",
            // "CD (2)",
            "CD (3)"
        ],
        "data": [
            {
                "success": true,
                "width": 762,
                "height": 757
```

Then replace Group Mug datas with mugMockup.



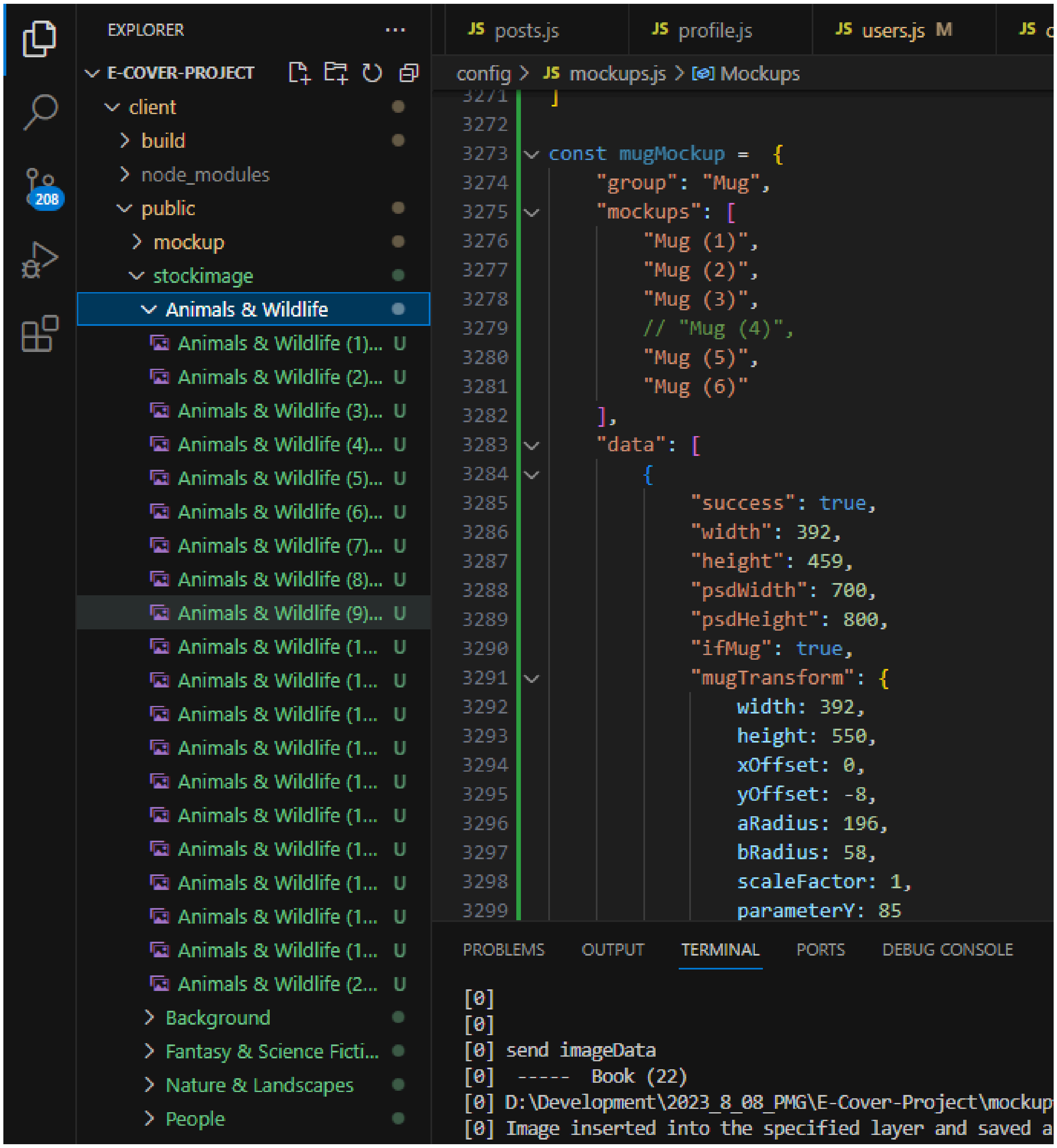
```
const Mockups = [
    {…
    },
    {…
    },
    {
        "group": "Mug",
        "mockups": [ …
        ],
        "data": [ …
        ]
    },
    {…
    },
    {…
    },
    {…
    },
    {…
    },
    {…
    },
    {…
    },
```



```
]

const mugMockup = {
    "group": "Mug",
    "mockups": [
        "Mug (1)",
        "Mug (2)",
        "Mug (3)",
        // "Mug (4)",
        "Mug (5)",
        "Mug (6)"
    ],
    "data": [
        {
            "success": true,
            "width": 392,
            "height": 459,
            "psdWidth": 700,
            "psdHeight": 800,
            "ifMug": true,
            "mugTransform": {
                width: 392,
                height: 550,
                xOffset: 0,
                yOffset: -8,
                aRadius: 196,
                bRadius: 58,
                scaleFactor: 1,
                parameterY: 85
```

```
[0]
[0]
[0] send imageData
[0]  -----  Book (22)
[0] D:\Development\2023_8_08_PMG\E-Cover-Project\mockupfiles\psd\Book (22).psd
```

7.2. Adding Stock Iamges


EXPLORER
E-COVER-PROJECT
client
build
node_modules
public
mockup
stockimage
Animals & Wildlife
Animals & Wildlife (1)... U
Animals & Wildlife (2)... U
Animals & Wildlife (3)... U
Animals & Wildlife (4)... U
Animals & Wildlife (5)... U
Animals & Wildlife (6)... U
Animals & Wildlife (7)... U
Animals & Wildlife (8)... U
Animals & Wildlife (9)... U
Animals & Wildlife (1... U
Animals & Wildlife (1... U
Animals & Wildlife (1... U
Animals & Wildlife (1... U
Animals & Wildlife (1... U
Animals & Wildlife (1... U
Animals & Wildlife (1... U
Animals & Wildlife (1... U
Animals & Wildlife (1... U
Animals & Wildlife (1... U
Animals & Wildlife (2... U
Background
Fantasy & Science Ficti...
Nature & Landscapes
People
208
JS posts.js
JS profile.js
JS users.js M
config > JS mockups.js > Mockups
PROBLEMS
OUTPUT
TERMINAL
PORTS
DEBUG CONSOLE

```
]

const mugMockup =  {
    "group": "Mug",
    "mockups": [
        "Mug (1)",
        "Mug (2)",
        "Mug (3)",
        // "Mug (4)",
        "Mug (5)",
        "Mug (6)"
    ],
    "data": [
        {
            "success": true,
            "width": 392,
            "height": 459,
            "psdWidth": 700,
            "psdHeight": 800,
            "ifMug": true,
            "mugTransform": {
                width: 392,
                height: 550,
                xOffset: 0,
                yOffset: -8,
                aRadius: 196,
                bRadius: 58,
                scaleFactor: 1,
                parameterY: 85
```

```
[0]
[0]
[0] send imageData
[0]  -----  Book (22)
[0] D:\Development\2023_8_08_PMG\E-Cover-Project\mockup
[0] Image inserted into the specified layer and saved a
```

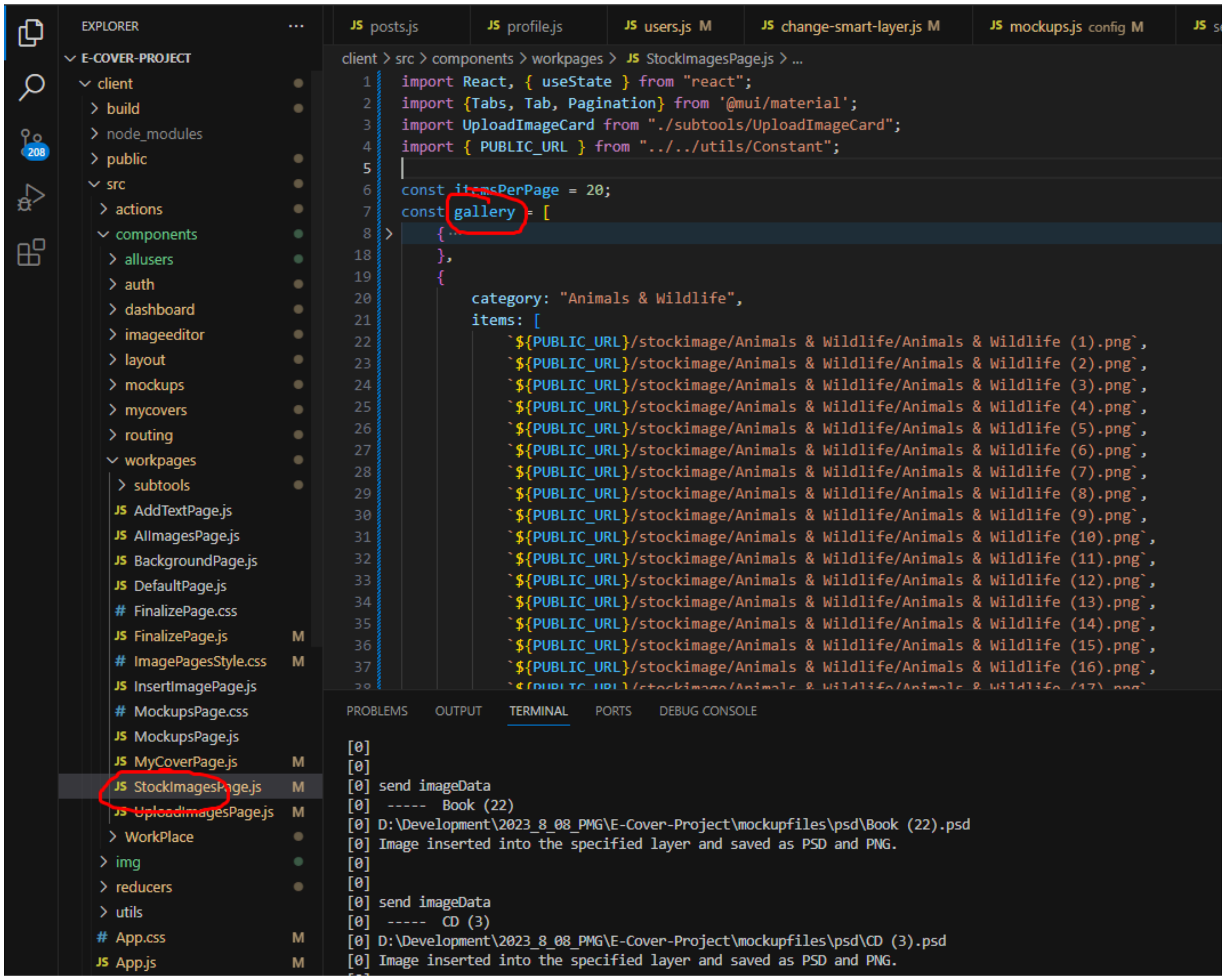
Saving PNG file

Path: client\public\stockimage\[Category name]\[file name]

ex: client\public\stockimage\Animals & Wildlife\Animals & Wildlife (1).png

Adding item to Stock Image page

Path: client\src\components\workpages\StockImagesPage.js



```
import React, { useState } from "react";
import {Tabs, Tab, Pagination} from '@mui/material';
import UploadImageCard from "./subtools/UploadImageCard";
import { PUBLIC_URL } from "../../utils/Constant";

const itemsPerPage = 20;
const gallery = [
    {…
    },
    {
        category: "Animals & Wildlife",
        items: [
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (1).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (2).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (3).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (4).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (5).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (6).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (7).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (8).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (9).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (10).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (11).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (12).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (13).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (14).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (15).png`,
            `${PUBLIC_URL}/stockimage/Animals & Wildlife/Animals & Wildlife (16).png`,
```

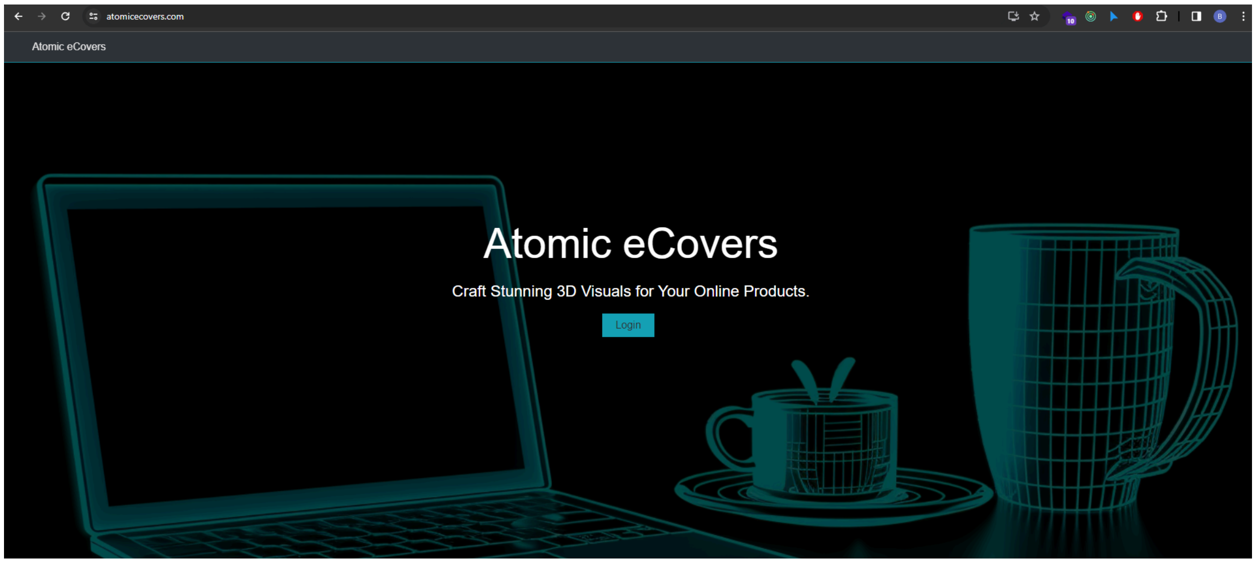
```
[0]
[0]
[0] send imageData
[0]  ----- Book (22)
[0] D:\Development\2023_8_08_PMG\E-Cover-Project\mockupfiles\psd\Book (22).psd
[0] Image inserted into the specified layer and saved as PSD and PNG.
[0]
[0]
[0] send imageData
[0]  ----- CD (3)
[0] D:\Development\2023_8_08_PMG\E-Cover-Project\mockupfiles\psd\CD (3).psd
[0] Image inserted into the specified layer and saved as PSD and PNG.
```

8. Front-end Design

6.1. Home page

url: /

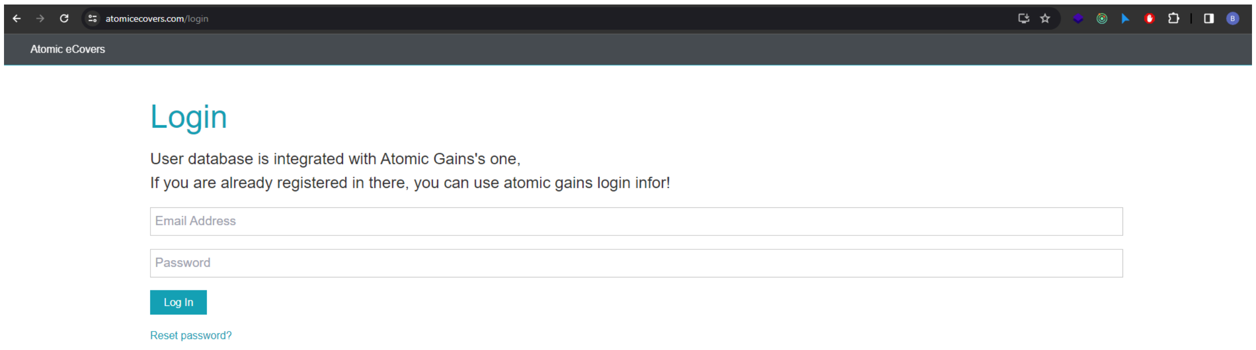
Component file: client\src\components\layout\Landing.js



6.2. Login page

url: /login

Component file: client\src\components\auth\Login.js

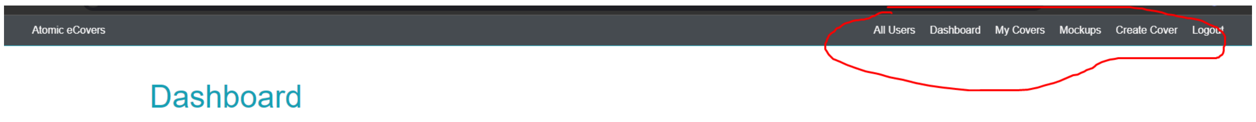


Login

User database is integrated with Atomic Gains's one,
If you are already registered in there, you can use atomic gains login infor!

6.3. Site Header

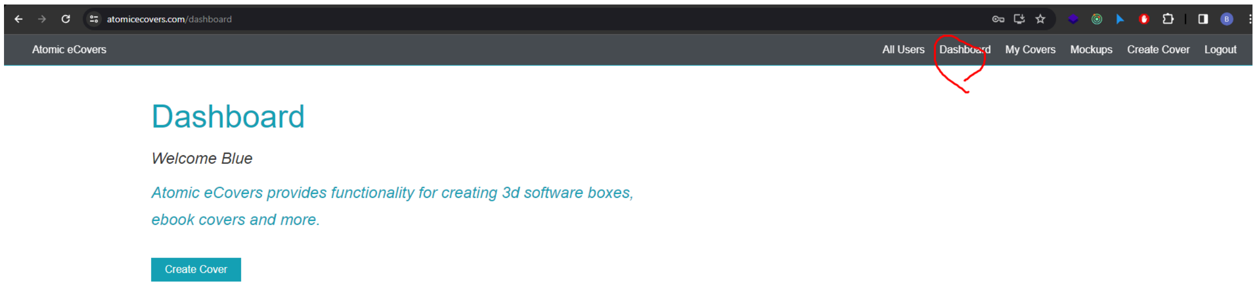
Component file: client\src\components\layout\Navbar.js



6.4. Dashboard Page

url: /dashboard

Component file: client\src\components\dashboard\Dashboard.js



Dashboard

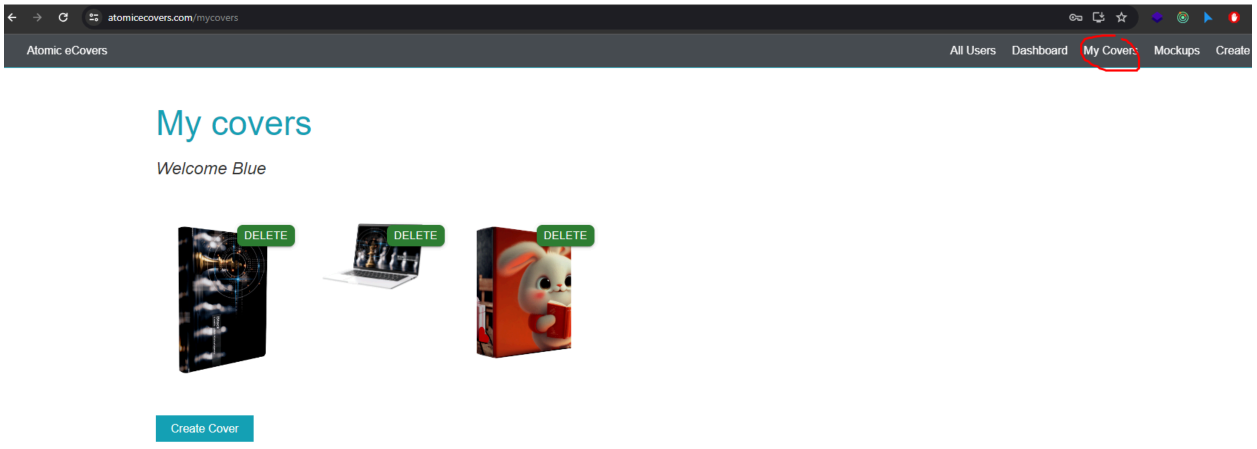
Welcome Blue

Atomic eCovers provides functionality for creating 3d software boxes, ebook covers and more.

6.5. My Covers Page

url: /mycovers

Component file: client\src\components\mycovers\Mycovers.js



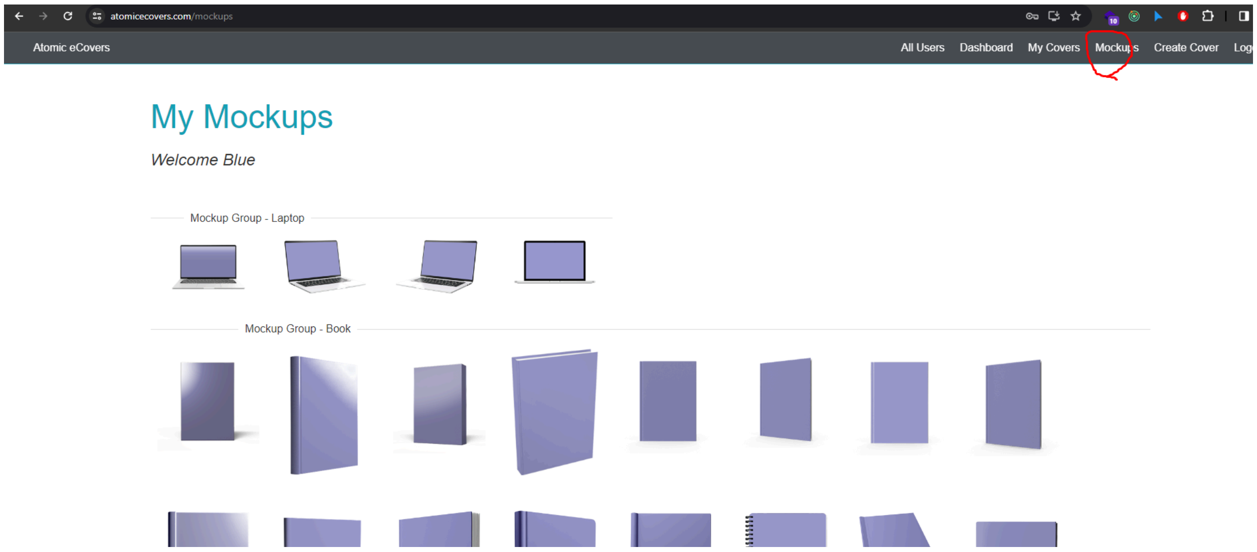
My covers

Welcome Blue

6.6. Mockups page

url: /mockups

Component file: client\src\components\mockups\Mockups.js



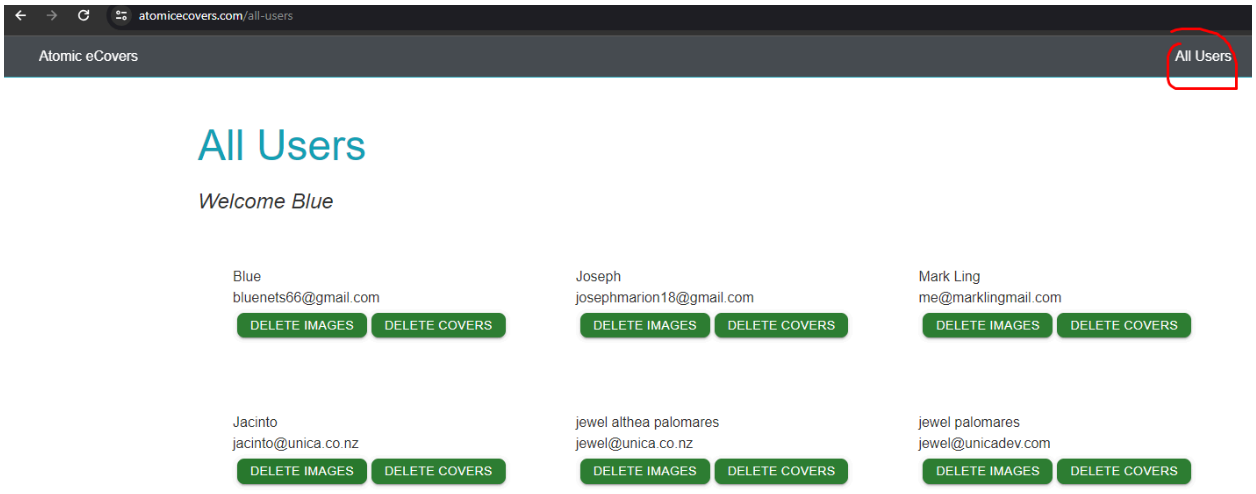
My Mockups

Welcome Blue

6.7. All users page

url: /all-users

Component file: client\src\components\allusers\AllUsers.js



All Users

Welcome Blue

Blue
bluenets66@gmail.com
DELETE IMAGES DELETE COVERS

Joseph
josephmarion18@gmail.com
DELETE IMAGES DELETE COVERS

Mark Ling
me@marklingmail.com
DELETE IMAGES DELETE COVERS

Jacinto
jacinto@unica.co.nz
DELETE IMAGES DELETE COVERS

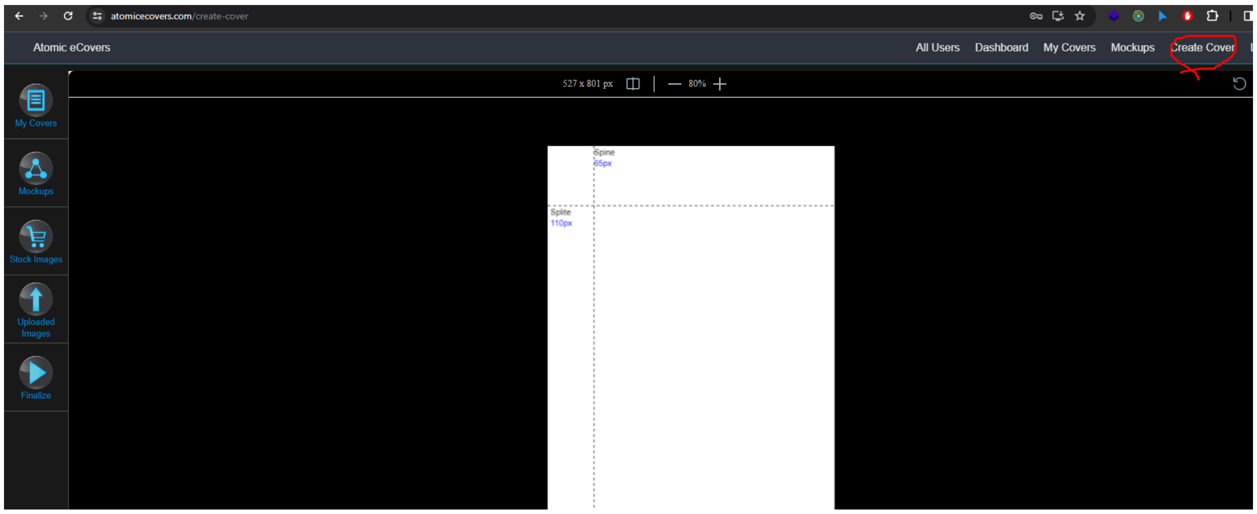
jewel althea palomares
jewel@unica.co.nz
DELETE IMAGES DELETE COVERS

jewel palomares
jewel@unicadev.com
DELETE IMAGES DELETE COVERS

6.8. Create Cover page

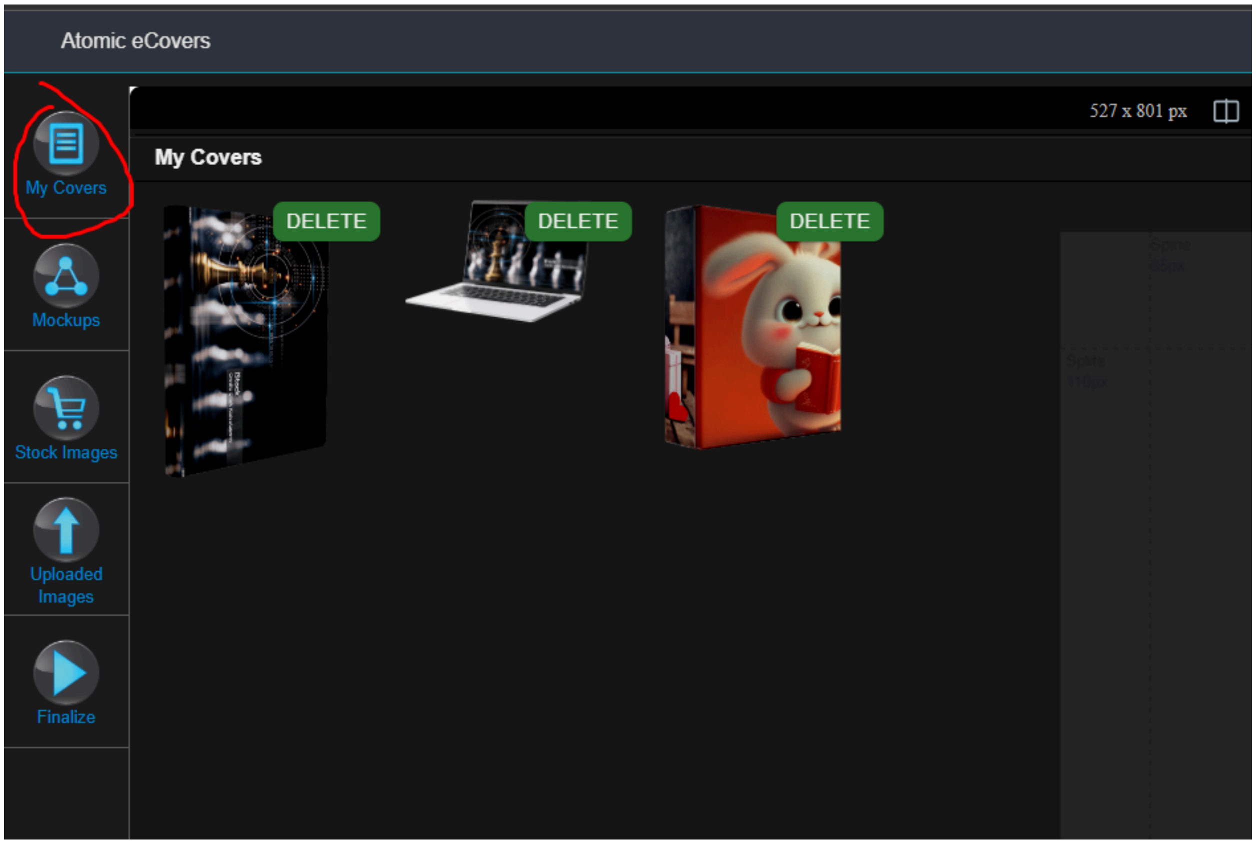
url: /create-cover

Component file: client\src\components\WorkPlace\CreateCover.js



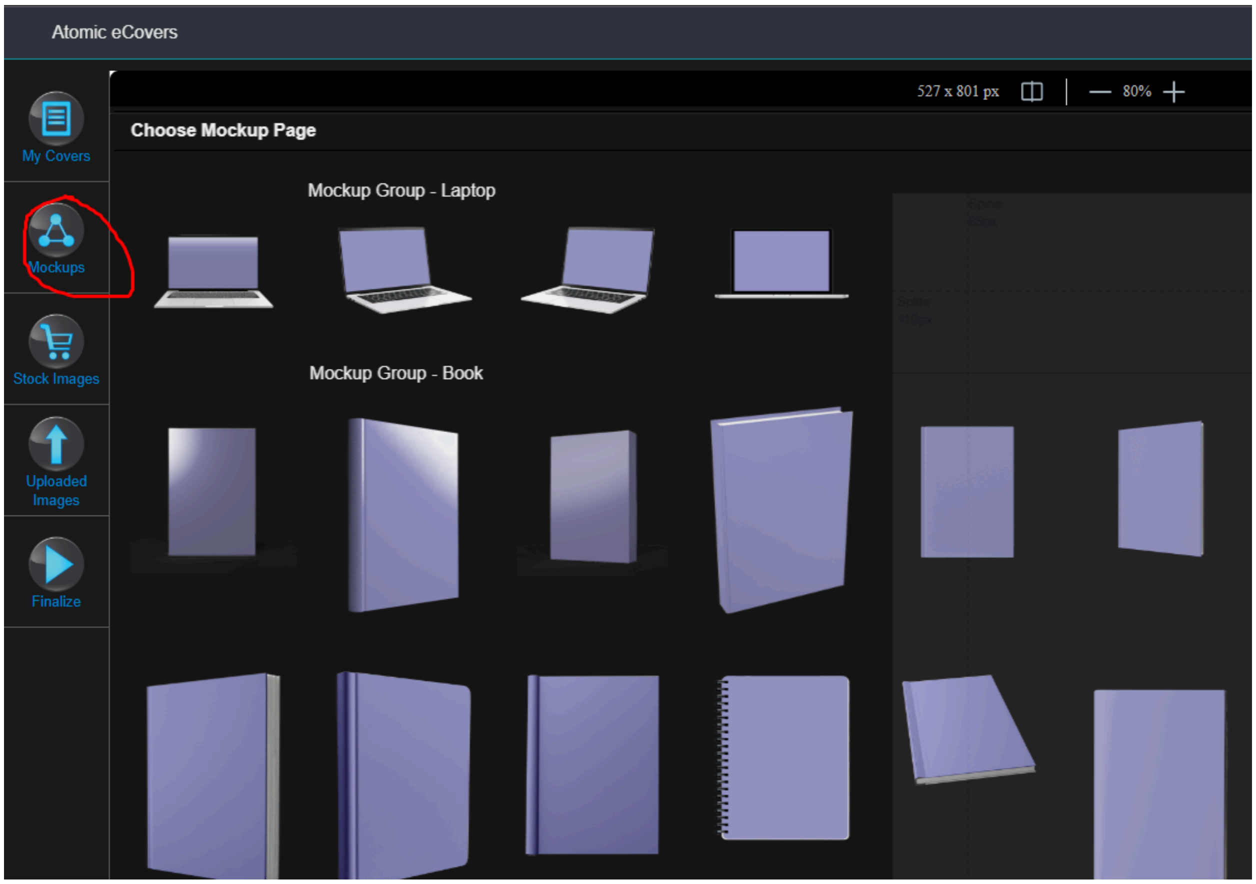
6.8.1. My Covers Component

File: client\src\components\workpages\MyCoverPage.js



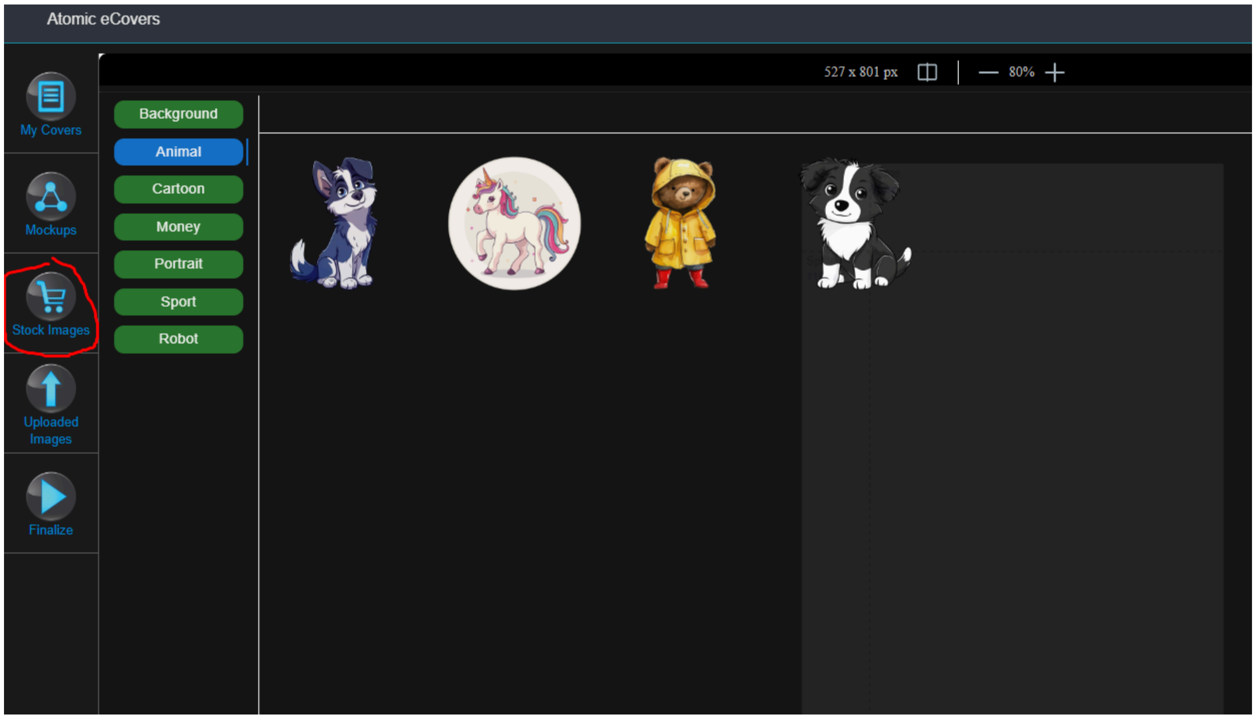
6.8.2. Mockups Component

File: client\src\components\workpages\MockupsPage.js



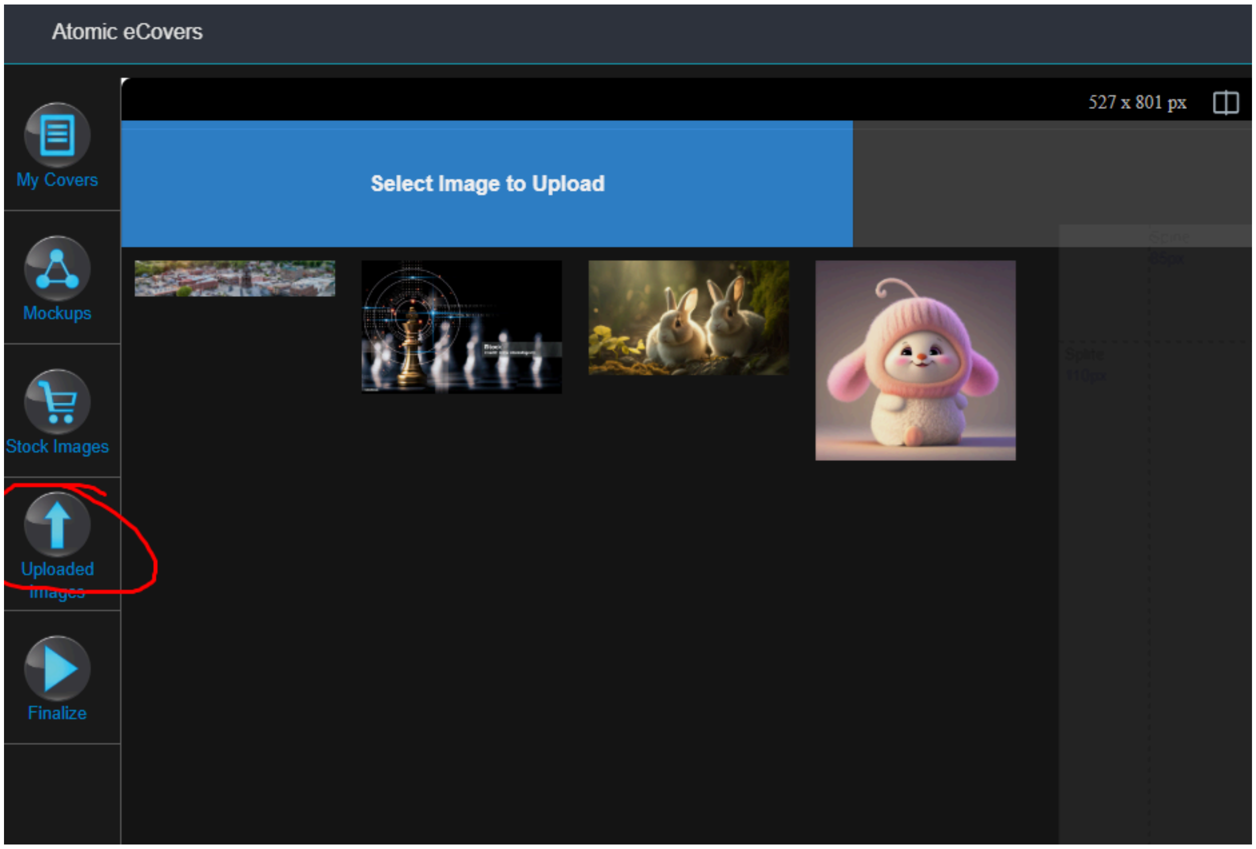
6.8.3. Stock Images Component

File: client\src\components\workpages\StockImagesPage.js



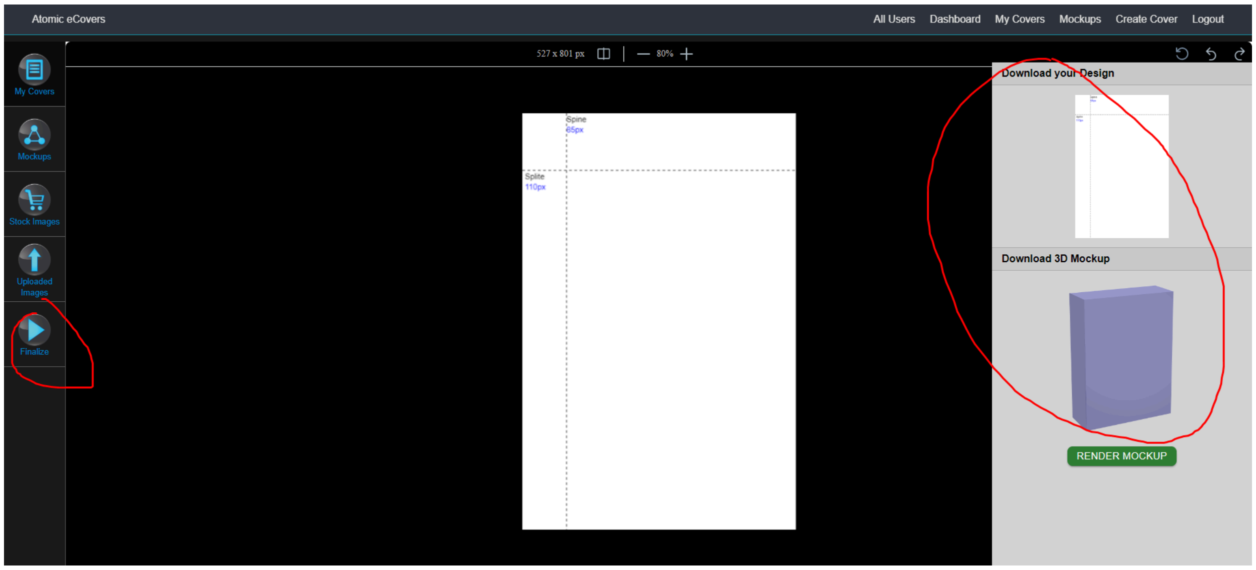
6.8.4. Uploaded Images Component

File: client\src\components\workpages\UploadImagesPage.js



6.8.5. Finalize Component

File: client\src\components\workpages\FinalizePage.js



7. Hosting

7.1. Running project dev version.

Install applications: Node v16.16.0, MongoDB v6.0, Python v3.11.5

Install python package, psd-tools: *run:* `pip install psd-tools`

Navigate to Project folder

run: `npm install`

run: `npm run server`

run: `cd client`

run: `npm install`

run: `npm start`

7.2. Project hosting on CentOS server.

7.2.1. Running Project

Install applications: Node v16.16.0, MongoDB v6.0, Python v3.11.5,

Install python package, psd-tools: *run:* `pip install psd-tools`

Copy and paste Project files to the server.

run: `npm install -g pm2`

run: `npm install -g serve`

Navigate to Project folder

run: `npm install`

run: `pm2 start npm --name "ecover-server" -- run server` *run:* `cd client`

run: `npm install`

run: `npm run build`

run: `cd build`

run: `pm2 serve . 3000 --name “ecover-front” -spa -s`

You can check pm2 processes: *run:* `pm2 list`

7.2.2. Setting up nginx proxy

Install nginx

Create /etc/nginx/conf.d/React.conf

```
server {
   listen 80;
   listen             443 ssl;
   server_name        atomicecovers.com;
   ssl_certificate     /etc/ssl/atomicecovers.com.crt;
   ssl_certificate_key /etc/ssl/atomicecovers.com.key;
   ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;
   ssl_ciphers         HIGH:!aNULL:!MD5;

   location / {
      proxy_set_header   X-Forwarded-For $remote_addr;
      proxy_set_header Host $host;
      proxy_pass         http://localhost:3000;
   }
```

```
    location /api {
        proxy_set_header   X-Forwarded-For $remote_addr;
        proxy_set_header Host $host;
        proxy_pass         http://localhost:5000;
    }
}
```

Create ssl crt and key files:

/etc/ssl/atomicecovers.com.crt

/etc/ssl/atomicecovers.com.key

Start nginx process

run: `systemctl start nginx`

Before starting nginx process, should stop httpd apache process