GitHub: https://github.com/adepge/ecm2434-exceptionhandlers

Backend files

database

admin.py

> backend/PostCard/database/admin.py

Written by Ziyad Alnawfal, Eugene Au, Jayant Chawla

```
from django.contrib import admin
from .models import Geolocation, Posts, Stickers, StickersUser, PostsUser
# Register models to admint page based on the models.py file, adding the
list_display and search_fields to make it easier to search and view the data
class PostsAdmin(admin.ModelAdmin):
    list_display = ('id', 'userid', 'datetime')
    search_fields = ['username__username']

class GeolocationAdmin(admin.ModelAdmin):
    list_display = ('id', 'location', 'latitude', 'longitude')

class StickersAdmin(admin.ModelAdmin):
    list_display = ('id', 'stickersName', 'stickersDescription', 'fileName')

class StickersUserAdmin(admin.ModelAdmin):
    list_display = ('id', 'username')

class PostUserAdmin(admin.ModelAdmin):
    display = ('userID')

# Register to admin site
admin.site.register(Geolocation, GeolocationAdmin)
admin.site.register(Posts, PostsAdmin)
admin.site.register(Stickers, StickersAdmin)
admin.site.register(StickersUser, StickersUserAdmin)
admin.site.register(PostsUser, PostUserAdmin)
```

apps.py

> backend/PostCard/database/apps.py

Written by Ziyad Alnawfal

```
from django.apps import AppConfig

class DatabaseConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'database'
```

models.py

> backend/PostCard/database/models.py

Written by Ziyad Alnawfal, Eugene Au, Jayant Chawla, Ben Ellison

```
from django.db import models
from django.contrib.auth.models import User #Importing django provided User model
with build in authentication

# Each FK is set to on_delete = models.CASCADE to uphold DB integrity and
consisentiy

class Geolocation(models.Model):
    id = models.AutoField(primary_key=True)
    location = models.CharField(max_length=255)
    latitude = models.FloatField(max_length=255)
    longitude = models.FloatField(max_length=255)
    latitude = models.FloatField(default=0.0)
    longitude = models.FloatField(default=0.0)
    def __str__(self):
        return f"{self.location} (Lat: {self.latitude}, Lng: {self.longitude})"

class Posts(models.Model):
    id    = models.AutoField(primary_key=True)
    image = models.ImageField(upload_to='media')
    geolocID = models.ForeignKey(Geolocation, on_delete = models.CASCADE)
    userid = models.ForeignKey(User, on_delete = models.CASCADE)
    caption  = models.CharField(max_length = 255)
    datetime = models.DateTimeField(auto_now_add = True) #Creates a timestamp

# the post user has collected
# {userid: 1, postids: [1,2,3,4,5]}
# TODO
class PostsUser(models.Model):
    userID = models.OneToOneField(User, on_delete = models.CASCADE)
    postID = models.ManyToManyField(Posts)

class Stickers(models.Model):
    id    = models.AutoField(primary_key=True)
    stickersName = models.CharField(max_length = 50)
```

```
    stickersDescription = models.CharField(max_length = 100)
    fileName = models.CharField(max_length = 100)

class StickersUser(models.Model):
    id = models.AutoField(primary_key=True)
    stickersID = models.ForeignKey(Stickers, on_delete = models.CASCADE)
    username   = models.ForeignKey(User, on_delete = models.CASCADE)
```

PostCard

asgi.py

> backend/PostCard/PostCard/asgi.py

Written by Ziyad Alnawfal

```
"""
ASGI config for PostCard project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/5.0/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'PostCard.settings')

application = get_asgi_application()
```

serializers.py

> backend/PostCard/PostCard/serializers.py

Written by Jayant Chawla

```
from rest_framework import serializers
from database.models import Geolocation, Posts, Stickers, StickersUser, PostsUser
from django.contrib.auth.models import User
```

```
# Add all the serializers here for the models
class GeolocationSerializer(serializers.ModelSerializer):
    class Meta:
        model = Geolocation
        fields = '__all__'  # This will include all fields from the Geolocation
model

class PostsSerializer(serializers.ModelSerializer):
    class Meta:
        model = Posts
        fields = '__all__'  # This will include all fields from the Posts model

class StickersSerializer(serializers.ModelSerializer):
    class Meta:
        model = Stickers
        fields = '__all__'  # This will include all fields from the Stickers model

class StickersUserSerializer(serializers.ModelSerializer):
    class Meta:
        model = StickersUser
        fields = '__all__'  # This will include all fields from the StickersUser
model

class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = '__all__'

class PostsUserSerializer(serializers.ModelSerializer):
    class Meta:
        model = PostsUser
        fields = '__all__'  # This will include all fields from the PostsUser model
```

settings.py

> backend/PostCard/PostCard/settings.py

Written by Ziyad Alnawfal, Eugene Au, Jayant Chawla

```
"""
Django settings for PostCard project.

Generated by 'django-admin startproject' using Django 5.0.2.

For more information on this file, see
https://docs.djangoproject.com/en/5.0/topics/settings/

For the full list of settings and their values, see
```

```
https://docs.djangoproject.com/en/5.0/ref/settings/
"""

from pathlib import Path
import os
from pathlib import Path

# settings.py


# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-$p9fq!f87crk&+doder0*xttbfp^-tn7nsw-ao#@umqe4fm&51'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition
# Included all the apps we add , and other dependiences
INSTALLED_APPS = [
    'posts',
    'database.apps.DatabaseConfig',
    'user_authentication',
    'rest_framework',
    'corsheaders',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'PostCard',
    'rest_framework.authtoken',
]

# configured the settings to allow us to use the restframe
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.SessionAuthentication',
        'rest_framework.authentication.BasicAuthentication',
```

```
        'rest_framework.authentication.TokenAuthentication',
    ],
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ]
}

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
]
#Allows frontend to connect to our server
CORS_ALLOWED_ORIGINS = [
    "http://localhost:5173", "http://localhost:5174", "http://localhost:5175",
    # 'corsheaders.middleware.CorsMiddleware',
    # 'django.middleware.common.CommonMiddleware',
]

CORS_ALLOW_CREDENTIALS = True

ROOT_URLCONF = 'PostCard.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'PostCard.wsgi.application'

# Database
# https://docs.djangoproject.com/en/5.0/ref/settings/#databases
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),

    }
}

# Password validation
# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/5.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

urls.py

backend/PostCard/PostCard/urls.py

Written by Ziyad Alnawfal, Jayant Chawla

```
from django.contrib import admin
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from django.conf import settings
from django.conf.urls.static import static

import posts.urls
# Add a default router to the urlpatterns which should be the main site for the API
router = DefaultRouter()

urlpatterns = [
    path('', include(router.urls)),
    path('api/',include(posts.urls)), # Contains all the API Endpoints
    path('admin/', admin.site.urls)
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

wsgi.py

backend/PostCard/PostCard/wsgi.py

Written by Ziyad Alnawfal

```
"""
WSGI config for PostCard project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/5.0/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'PostCard.settings')

application = get_wsgi_application()
```

---

posts

apps.py

> backend/PostCard/posts/apps.py

Written by Ziyad Alnawfal

```
from django.apps import AppConfig

class PostsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'posts'
```

urls.py

> backend/PostCard/posts/urls.py

Written by Ziyad Alnawfal, Eugene Au, Jayant Chawla

```
from django.urls import path
from .views import *
from user_authentication.views import *
from django.conf.urls.static import static
from django.conf import settings


urlpatterns = [
    #USER AUTHENTICATION API ENDPOINT
    path('register/', UserRegisterAuthentication, name="register"),
    path('login/',UserLoginAuthentication, name='login'),
    #-----
    #POSTS API ENDPOINT
    path('posts/', PostsList.as_view(), name='posts-list'),
    path('createPost/', createPost , name='create-post'),
    path('posts/<int:pk>/', PostsDetail.as_view(), name='posts-detail'),
    path('geolocations/', GeolocationList.as_view(), name='geolocation-list'),
    path('geolocations/<int:pk>/', GeolocationDetail.as_view(), name='geolocation-
detail'),
    path('getRecentPosts/', getPostsLast24Hours, name='get-recent-posts'),
    #----
    path('posts/recent/', getPostsLast24Hours, name='posts-recent'),
    # STICKER API ENDPOINT
    path('stickers/', StickersList.as_view(), name='sticker-list'),
    path('stickers/<int:pk>/', StickersDetail.as_view(), name='sticker-detail'),
    path('stickerusers/', StickersUserList.as_view(), name='stickeruser-list'),
    path('stickerusers/<int:pk>/', StickersUserDetail.as_view(), name='stickeruser-
detail'),
```

```
    #---

    #POSTUSER API ENDPOINT
    path('collectPost/', addCollection, name='postuser-list'),
    path('collectedPosts/', getCollections, name='collected-posts'),

    #USER API ENDPOINT
    path('getUser/', getUser, name='user-list'),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

views.py

> backend/PostCard/posts/views.py

Written by Ziyad Alnawfal, Eugene Au, Jayant Chawla

```
from django.shortcuts import render
from rest_framework import generics
from database.models import Geolocation, Posts, Stickers, StickersUser, PostsUser
from PostCard.serializers import PostsSerializer, GeolocationSerializer,
StickersSerializer, StickersUserSerializer, PostsUserSerializer
from rest_framework.permissions import AllowAny
from rest_framework.decorators import permission_classes
from rest_framework.decorators import api_view
from rest_framework.response import Response
from rest_framework import status

# creating the views based on the models, should be able to list and view the data

#Returns All posts objects
class PostsList(generics.ListCreateAPIView):
    queryset = Posts.objects.all()
    serializer_class = PostsSerializer
    permission_classes = [AllowAny]

#Returns a specific post made by using the postID
class PostsDetail(generics.RetrieveUpdateDestroyAPIView):
    queryset = Posts.objects.all()
    serializer_class = PostsSerializer
    permission_classes = [AllowAny]

#Returns all geolocations retrieved from posts made
class GeolocationList(generics.ListCreateAPIView):
    queryset = Geolocation.objects.all()
    serializer_class = GeolocationSerializer
```

```
    permission_classes = [AllowAny]

#Returns a specific geolocation
class GeolocationDetail(generics.RetrieveAPIView):
    queryset = Geolocation.objects.all()
    serializer_class = GeolocationSerializer
    permission_classes = [AllowAny]
#Returns all stickers made
class StickersList(generics.ListCreateAPIView):
    queryset = Stickers.objects.all()
    serializer_class = StickersSerializer
    permission_classes = [AllowAny]

#Returns a specific sticker made
class StickersDetail(generics.RetrieveAPIView):
    queryset = Stickers.objects.all()
    serializer_class = StickersSerializer
    permission_classes = [AllowAny]

#Returns all stickers owned by a user
class StickersUserList(generics.ListCreateAPIView):
    queryset = StickersUser.objects.all()
    serializer_class = StickersUserSerializer
    permission_classes = [AllowAny]

#Returns a sticker owned by a user
class StickersUserDetail(generics.RetrieveAPIView):
    queryset = StickersUser.objects.all()
    serializer_class = StickersUserSerializer
    permission_classes = [AllowAny]

#Returns all posts made by a user
class PostUser(generics.ListCreateAPIView):
    queryset = PostsUser.objects.all()
    serializer_class = PostsUserSerializer
    permission_classes = [AllowAny]

@api_view(['POST']) # Secuirty purposes we do not want to append user details to
header
@permission_classes([AllowAny])
def createPost(request):
    try:
        userid = request.user.id
    except:
        # if user is not logged in, then raise an error
        return Response({"message":"User not logged
in"},status=status.HTTP_400_BAD_REQUEST)

    #Getting the length of image name
    filename = request.FILES['image'].name
```

```
    filename_length = len(filename)

    #If its larger than 100 than get upload the last 30 chars to avoid errors
    if filename_length > 100:
        request.FILES['image'].name = filename[-30:]

    # Create a mutable copy of request.data
    data = request.data.copy()
    print(data['image'])
    print(data)
    data['userid'] = userid  # Add 'userid' key

    print(data)

    # Create a serializer instance with the mutable copy of request.data
    serialized = PostsSerializer(data=data)

    if serialized.is_valid():
        serialized.save()
        return Response({"message":"Post made"},status=status.HTTP_201_CREATED) #
Successful post creation
    else:
        return Response(status=status.HTTP_400_BAD_REQUEST) # Failed post creation

@api_view(['POST' ,'Get'])
@permission_classes([AllowAny])
def getUser(request):
    try:
        userid = request.user.id
        username = request.user.username
    except:
        # if user is not logged in, then raise an error
        return Response({"message":"User not logged
in"},status=status.HTTP_400_BAD_REQUEST)
    return Response({"userid":userid,"username":username},status=status.HTTP_200_OK)
# Successful user creation


from django.utils import timezone
from datetime import timedelta

@api_view(['POST','GET'])
@permission_classes([AllowAny])
def getPostsLast24Hours(request):
    try:
        current_time = timezone.now()

        #stores the last 24 hrs
        time_24_hours_ago = current_time - timedelta(days=1)
```

```
        #Stores posts made in the last 24hrs
        recent_posts = Posts.objects.filter(datetime__range=[time_24_hours_ago,
current_time]).select_related('geolocID')

        data = []
        # Prepare the data to return
        for post in recent_posts:
            data.append({
            'id': post.id,
            'image': post.image.url,
            'caption': post.caption,
            'datetime': post.datetime,
            'open': False,
            'position': {
                'location': post.geolocID.location,
                'lat': post.geolocID.latitude,
                'lng': post.geolocID.longitude,
                }
            }
            )


            # Return the data as JSON
        return Response(data, status=status.HTTP_200_OK)

    except Exception as e:
        return Response({"error": str(e)}, status=status.HTTP_400_BAD_REQUEST)



@api_view(['POST'])
@permission_classes([AllowAny])
def addCollection(request):
    try:
        user = request.user
    except Exception as e:
        #if user is not logged in
        return Response({"message": str(e)}, status=status.HTTP_400_BAD_REQUEST)

    post_id = request.data.get('postid')
    if not post_id:
        #If post not found
        return Response({"message": "postid not provided"},
status=status.HTTP_400_BAD_REQUEST)

    try:
        # Retrieve the Post instance using post_id
        post = Posts.objects.get(id=post_id)
    except Posts.DoesNotExist:
        return Response({"message": "Post does not exist"},
```

```
status=status.HTTP_404_NOT_FOUND)

    # Get or create a PostsUser instance for the user
    posts_user, created = PostsUser.objects.get_or_create(userID=user)

    # Add the post to the user's collection
    posts_user.postID.add(post)

    return Response({"message": "Post added to collection"},
status=status.HTTP_201_CREATED)

@api_view(['POST'])
@permission_classes([AllowAny])
#returns the collection of posts saved by a user
def getCollections(request):
    try:
        user = request.user
    except Exception as e:
        return Response({"message": str(e)}, status=status.HTTP_400_BAD_REQUEST)

    try:
        posts_user = PostsUser.objects.get(userID=user)

    except PostsUser.DoesNotExist:
        posts_user = []

    print(posts_user)
    serializer = PostsUserSerializer(posts_user)
    postlist = serializer.data.get('postID')
    posts = Posts.objects.filter(id__in=postlist)
    serializer = PostsSerializer(posts, many=True)

    return Response(serializer.data, status=status.HTTP_200_OK)
```

---

user_authentication

apps.py

> backend/PostCard/user_authentication/apps.py

Written by Ziyad Alnawfal

```
from django.apps import AppConfig

class UserAuthenticationConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'user_authentication'
```

forms.py

> backend/PostCard/user_authentication/forms.py

Written by Ziyad Alnawfal

```
from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm

# inherting the usercreationform because it provides built in authentication
class UserRegisteration(UserCreationForm):
    email = forms.EmailField()
    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']
```

models.py

> backend/PostCard/user_authentication/models.py

Written by Ziyad Alnawfal and Eugene Au

```
from django.db import models

# Create your models here.
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin as BaseUserAdmin
from django.contrib.auth.models import User

class UserAdmin(BaseUserAdmin):
    # Customize list_display to show the desired fields
    list_display = ('id', 'username', 'email', 'first_name', 'last_name',
'is_staff')
    # You can also customize other options like list_filter, search_fields etc.

# Unregister the original User admin and register the customized version
admin.site.unregister(User)
admin.site.register(User, UserAdmin)
```

serializers.py

> backend/PostCard/user_authentication/serializers.py

Written by Ziyad Alnawfal and Eugene Au

```
from rest_framework import serializers
from django.contrib.auth.models import User

# Converting the incoming data type into django model to save to the database,
cannot save json data
class UserRegisterSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['id', 'username','password','email'] # all reveleant fields

class UserLoginSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['username','password']
```

views.py

> backend/PostCard/user_authentication/views.py

Written by Ziyad Alnawfal and Eugene Au

```
from rest_framework import serializers
from django.contrib.auth.models import User

# Converting the incoming data type into django model to save to the database,
cannot save json data
class UserRegisterSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['id', 'username','password','email'] # all reveleant fields

class UserLoginSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['username','password']
```

Frontend files

App.jsx

> frontend/src/App.jsx

Written by Eugene Au

```
import { Routes, Route } from "react-router-dom";
import Header from "./features/header";
import Footer from "./features/footer";

import MapPage from "./pages/map";
import FeedPage from "./pages/feed";
import Capture from "./pages/capture";
import RegisterPage from "./pages/register";
import LoginPage from "./pages/login";
import PageNoFound from "./pages/pageNoFound";
import ProfilePage from "./pages/profilepage";
import Test from "./pages/test";

function App() {

  return (
    <>
      <Header />
      <Routes>

        <Route path="/register" element={<RegisterPage />} />
        <Route path="/login" element={<LoginPage />} />
        <Route path="/" element={<MapPage />} />
        <Route path="/feed" element={<FeedPage />} />
        <Route path="/capture" element={<Capture />} />
        <Route path="/profile" element={<ProfilePage />} />
        <Route path="*" element={<PageNoFound />} />
        <Route path="/test" element={<Test />} />
      </Routes>
      <Footer />
    </>
  );
}
export default App;
```

index.css

> frontend/src/index.css

Written by Adam George

```
@import url("https://fonts.googleapis.com/css2?
family=Outfit:wght@100;200;300;400;500;600;700;800;900&display=swap");
```

```
:root {
  font-family: Outfit;
  line-height: 1.5;
  font-weight: 400;

  color-scheme: light dark;
  color: rgba(255, 255, 255, 0.87);
  background-color: white;

  font-synthesis: none;
  text-rendering: optimizeLegibility;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;

  --primary: #00DCA5;
  --grateful: #35B39A;
  --happy: #38C3EF;
  --anxious: #F89970;
  --tired: #927AF5;
  --down: #78819F;
  --none: #428BCA;
}

body {
  margin: 0;
}
/* Width of the scrollbar */
::-webkit-scrollbar {
  width: 0px;
}
```

main.jsx

> frontend/src/main.jsx

Generated by React, edited by Adam George

```
import React from "react";
import { createRoot } from "react-dom/client";
import { BrowserRouter } from "react-router-dom";
import App from "./App"; // Your main App component
import "./index.css";

const container = document.getElementById("root");
const root = createRoot(container);
root.render(
  <BrowserRouter>
    <App />
```

```
    </BrowserRouter>
);
```

Features components

CheckLogin.jsx

> frontend/src/features/CheckLogin.jsx

Written by Eugene Au

```
// Code to check if the user is logged in or not

import axios from "axios";
import Cookies from "universal-cookie";
import { useNavigate } from "react-router-dom";

const cookies = new Cookies();

async function CheckLogin() {

    if (cookies.get('token') === undefined) {
        window.location.href = "/login";
        return false;
    }

    try {
        const response = await axios.get(
            "http://127.0.0.1:8000/api/getUser/"
            ,
            {
                headers: {
                    "Content-Type": "multipart/form-data",
                    "Authorization": `Token ${cookies.get('token')}`, // Assuming
postData.username is the token
                },
            }
        );
        return true;
    } catch (error) {
        if (error.response) {
            if (error.response.data.detail === "Invalid token.") {
                cookies.remove('token');
                console.log("not logged in");
                window.location.href = "/login";
                return false
            } else {
```

```
                console.log("Response data:", error.response.data);
                console.log("Response status:", error.response.status);
                console.log("Response headers:", error.response.headers);
            }
        }
    }

}

export default CheckLogin;
```

DrawerDown.jsx

> frontend/src/features/DrawerDown.jsx

Written by Adam George

```
import { useState, useEffect, useRef } from "react";
import { useCollectedPinStore } from "../stores/pinStore";
import Polaroid from "./polaroid";
import handle from "../assets/map/handle.svg";
import "../styles/drawer-down.css";

function DrawerDown({ id, image, caption, drawerVisible, setDrawerVisible,
handleSubmit, handleClickPolaroid }) {
  const elementRef = useRef(null);
  const [closing, setClosing] = useState(drawerVisible);
  const [collected, setCollected] = useState(false);

  const collectedPins = useCollectedPinStore((state) => state.pinIds);

  useEffect(() => {
    if (collectedPins.includes(id)) {
      setCollected(true);
    }
  }, [collectedPins, id]);

  useEffect(() => {
    if (drawerVisible) {
      setTimeout(() => {
        document.addEventListener("click", handleClick);
      }, 650);
    }
  }, [drawerVisible]);

  const handleClick = (event) => {
    // Check if the clicked element is outside the element
```

```
    event.preventDefault();

    if (elementRef.current && !elementRef.current.contains(event.target)) {
      setClosing(true);
      document.removeEventListener("click", handleClick);
      setTimeout(() => {
        setDrawerVisible(false);
        setClosing(false);
      }, 650);
    }
  };

  const drawerClass = closing
    ? "drawer-container"
    : "drawer-container drawer-exit-top";

  return (
    <>
      <div className="drawer-wrapper">
        {drawerVisible && (
          <div className={drawerClass} ref={elementRef}>
            <div id="texture">
              <div id="polariod-container" onClick={handleClickPolaroid}>
                <Polaroid id="collect-polaroid" src={image} rotation={-5} caption=
{caption} />
              </div>
            </div>
            <button disabled={collected} onClick={handleSubmit}>{collected ? "Pin
collected" : "Add to collection"}</button>
            <img id="handle" src={handle}></img>
          </div>
        )}
      </div>
    </>
  );
}

export default DrawerDown;
```

footer.jsx

> frontend/src/features/footer.jsx

Written by Eugene Au

```
import React from "react";
import mapicon from "../assets/footer/map.svg";
import feedicon from "../assets/footer/feed.svg";
```

```
import topicon from "../assets/footer/top.svg";
import captureicon from "../assets/footer/capture.svg";
import { useLocation } from "react-router-dom";

import "./stylesheets/footer.css";
import { Link } from "react-router-dom";

export default function Footer() {
  const location = useLocation();

  return (
    <footer>
      {/* map icon */}
      <div className="button">
        {/* send user to the map page on click */}
        <Link to="/">
          <div
            className="backdrop"
            style={{
              background: location.pathname === "/" ? "#00DCA5" : "none",
            }}
          >
            <img src={mapicon} id="map-icon" alt="map-icon" />
          </div>
          <div className="footer-text">map</div>
        </Link>
      </div>

      {/* top icon */}
      <div className="button">
        {/* send user to the top page on click */}
        <Link to="/top">
          {/* if the user is on the top page, add backdrop to the icon */}
          <div
            className="backdrop"
            style={{
              background: location.pathname === "/top" ? "#00DCA5" : "none",
            }}
          >
            <img src={topicon} id="top-icon" alt="top-icon" />
          </div>
          <div className="footer-text">top</div>
        </Link>
      </div>

      {/* feed icon */}
      <div className="button">
        {/* send user to the feed page on click */}
        <Link to="/feed">
          <div
```

```
            className="backdrop"
            style={{
              background: location.pathname === "/feed" ? "#00DCA5" : "none",
            }}
          >
            <img src={feedicon} id="feed-icon" alt="feed-icon" />
          </div>
          <div className="footer-text">Collection</div>
        </Link>
      </div>

      {/* capture icon */}
      <div className="button">
        {/* send user to the capture page on click */}
        <Link to="/capture">
          <div
            className="backdrop"
            style={{
              background: location.pathname === "/capture" ? "#00DCA5" : "none",
            }}
          >
            <img src={captureicon} id="capture-icon" alt="capture-icon" />
          </div>
          <div className="footer-text">capture</div>
        </Link>
      </div>
    </footer>
  );
}
```

Geolocation.jsx

> frontend/src/features/Geolocation.js

Written by Adam George

```
function Geolocation(latitude, longitude, setLocation) {
  const apiKey = import.meta.env.VITE_GOOGLE_MAPS_API_KEY;
  fetch(`https://maps.googleapis.com/maps/api/geocode/json?
latlng=${latitude},${longitude}&key=${apiKey}`)
  .then(response => response.json())
  .then(data => {
    if (data.results && data.results.length > 0) {
      const placeName = data.results[0].formatted_address;
      setLocation(placeName.split(',')[0]);
    } else {
      setLocation('Unknown Location');
    }
```

```
    })
    .catch(error => {
      console.error(error);
    });
  }

  export default Geolocation;
```

header.jsx

> frontend/src/features/header.jsx

Written by Eugene Au

```
import React from "react";
import logo from "../assets/logo-notext.png";
import usericon from "../assets/header/user-icon.jpg";
import "./stylesheets/header.css";

function Header() {
  return (
    <header>
      <div id="header-wrapper">
        {/* the logo which is the title and the logo */}
        <div id="logo">
          <img src={logo} id="icon" alt="icon" height="30px" />
          <div id="title">Post-i-tivity</div>
        </div>

        {/* the user icon */}
        <img src={usericon} id="user-icon" alt="user-icon" width={"25px"} height=
{"25px"} style={{ border: "none", borderRadius: "25px" }} />
      </div>
    </header>
  );
}

export default Header;
```

InitMap.jsx

> frontend/src/features/InitMap.jsx

Written by Adam George

```
import logo from '../assets/logo-text.png'
import './stylesheets/InitMap.css'
import loadingtips from '../assets/loading/loadingtips.json'

const InitMap = ({ progress }) => {
    const randomTip = loadingtips.loadTips[Math.floor(Math.random() *
loadingtips.loadTips.length)]
    return (
        <div className={progress >= 100 ? "splash fade-out" : "splash"}>
            <img src={logo}></img>
            <div className="loading-bar">
                <div className="loading-bar-progress" style={{ width: `${progress >
100 ? 100 : progress}%` }} />
            </div>
            <div className="tips-wrapper">{randomTip}</div>
        </div>
     );
}

export default InitMap;
```

interactives.jsx

> frontend/src/features/Interactives.jsx

Written by Eugene Au

```
/* the interactives component is a child component of the card component. It
contains the location and share icons. */

import "./stylesheets/interactives.css";

import Location from "../assets/location.svg";
import Share from "../assets/foward.svg";

function Interactives({ location, userIcon }) {
  return (
    <div className="interactives">
      <div className="location element">
        <img src={Location} />
        {location}
      </div>
      <div className="share element">
        <img src={Share} />
      </div>
    </div>
  );
```

```
}

export default Interactives;
```

interactivesTop.jsx

> frontend/src/features/InitMap.jsx

Written by Eugene Au

```
/* the top bar of the interactives page, with the exit button */
import "./stylesheets/interactives.css";
import exit from "../assets/exit.svg";

function InteractivesTop() {
  return (
    <div className="interactives" style={{ flexDirection: "row-reverse" }}>
      <img src={exit} />
    </div>
  );
}

export default InteractivesTop;
```

loadingScreen.jsx

> frontend/src/features/loadingScreen.jsx

Written by Eugene Au

```
// a loading screen that will be displayed when the app is loading

import React from "react";
import "./stylesheets/loadingScreen.css";

function LoadingScreen({ active }) {
  return (
    <div
      className="loading-screen"
      style={active ? { display: "block" } : { display: "none" }}
    >
      <div className="loading-spinner"></div>
    </div>
  );
}
```

```
export default LoadingScreen;
```

MoodPrompt.jsx

> frontend/src/features/MoodPrompt.jsx

Written by Adam George

```
import close from '../assets/map/close.svg';

function MoodPrompt({onClickFunction}) {
    return (
      <div className="mood-prompt">
        <div id="mood-prompt-card">
          <img onClick={() => onClickFunction('none')} src={close}/>
          <p>How are you feeling today?</p>
          <button onClick={() => onClickFunction('grateful')} style=
{{backgroundColor: "var(--grateful)"}}>Grateful</button>
          <button onClick={() => onClickFunction('happy')} style={{backgroundColor:
"var(--happy)"}}>Happy</button>
          <button onClick={() => onClickFunction('anxious')} style=
{{backgroundColor: "var(--anxious)"}}>Anxious</button>
          <button onClick={() => onClickFunction('tired')} style={{backgroundColor:
"var(--tired)"}}>Tired</button>
          <button onClick={() => onClickFunction('down')} style={{backgroundColor:
"var(--down)"}}>Down</button>
          <div className="urgent-help">I need urgent help</div>
        </div>
      </div>
    );
  }

export default MoodPrompt;
```

polaroid.jsx

> frontend/src/features/polaroid.jsx

Written by Eugene Au

```
/* a polaroid component that takes in a src and a function to be called when
clicked. It also takes in a rotation value to rotate the polaroid. */

import "./stylesheets/polaroid.css";
```

```
function Polaroid({ src, func, rotation, caption}) {
  return (
    <div
      className="polaroid"
      style={{ transform: `rotate(${rotation}deg)` }}
      onClick={func}
    >
      <div className="padding">
        <img src={src} alt="polaroid" style={{ width: "100%" }} />
        <div className="caption">
          {caption}
        </div>
      </div>
    </div>
  );
}

export default Polaroid;
```

PostView.jsx

> frontend/src/features/PostView.jsx

Written by Eugene Au

```
/* a absolute view for the post */
import "./stylesheets/postView.css";

import Interactives from "../features/interactives";
import InteractivesTop from "../features/interactivesTop";

import Polaroid from "./polaroid";

function PostView({ caption, image, isActive, leaveFunction, location, userIcon }) {
  return (
    <div>
      <div
        className={isActive ? "display active" : "display"}
        onClick={leaveFunction}
      >
        <div className="post-wrapper">
          <div className="post">
            <InteractivesTop />
            <div className="image">
              <Polaroid src={image} caption={caption} />
            </div>
            <Interactives location={location} isActive={isActive} />
```

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

export default PostView;
```

footer.css

> frontend/src/features/stylesheets/footer.css

Written by Eugene Au

```
footer {
  /* make it be in the bottom all the time */
  position: fixed;
  bottom: 0;
  width: 100%;

  /* white background */
  background-color: white;

  /* evenly distribute the icons */
  display: flex;
  justify-content: space-evenly;
  align-items: center;
  padding: 10px 0px;

  /* text style */
  font-size: 10px;
  font-weight: 700;
  color: black;
  text-decoration: none;
}

footer a {
  /* remove the highlight */
  text-decoration: none;

  /* same color as before */
  color: inherit;

  /* prevent user fron selecting the text */
  user-select: none;
}
```

```
/* the backdrop */
.backdrop {
  /* align items in the center */
  display: flex;
  align-items: center;
  justify-content: center;

  /* round conoer */
  border: none;
  border-radius: 30px;

  /* space between that and the text */
  margin-bottom: 2.5px;

  /* animation */
  transition: background 0.2s ease-in-out;
}

footer .button {
  width: 55px;
}

.footer-text {
  text-align: center;
}
```

header.css

> frontend/src/features/stylesheets/header.css

Written by Eugene Au

```
header {
  /* absolute top */
  position: fixed;
  top: 0;
  width: 100%;
  z-index: 10;

  /* black background */
  background-color: black;
}

/* add space between logo and user icon */
#header-wrapper {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
```

```
  padding: 10px 15px;
}

/* center the logo */
#logo {
  display: flex;
  flex-direction: row;
  align-content: center;
  justify-content: center;
}

/* the text */
#logo #title {
  font-size: 15px;
  font-weight: 700;
  display: flex;
  align-items: center;
  justify-content: center;
}

/* the icon */
#icon {
  margin-right: 10px;
}
```

InitMap.css

> frontend/src/features/stylesheets/InitMap.css

Written by Adam George

```
.splash{
    position: absolute;
    height: 100vh;
    width: 100vw;
    background-color: white;
    z-index: 9999;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}

.splash img {
    width: 250px;
}

.loading-bar {
```

```
    margin-top: 50px;
    width: 80vw;
    max-width: 320px;
    height: 10px;
    background-color: #f3f3f3;
    border-radius: 10px;
    overflow: hidden;
}

.loading-bar-progress {
    height: 100%;
    background-color: #00DCA5;
    transition: width 0.5s;
}

.tips-wrapper{
    animation: fadeInLoading 0.5s ease-out;
    margin-top: 93px;
    color: #9F9F9F;
    height: 50px;
    width: 300px;
    text-align: center;
}

.fade-out {
    opacity: 0;
    transition: opacity 0.5s;
    transition-delay: 2s;
}

@keyframes fadeInLoading {
    from {
      transform: scale(0.8);
      opacity: 0;
    }
    to {
      opacity: 1;
      transform: scale(1);
    }
}
```

interactives.css

> frontend/src/features/stylesheets/interactives.css

Written by Eugene Au

```css
.interactives {
  /* the container */
  height: 35px;
  width: 100%;

  /* space between elements */
  display: flex;
  justify-content: space-between;
  align-items: center;

  /* padding on top*/
  margin-top: 10px;

  overflow: visible;

  /* font styles */
  font-size: 12px;
  font-weight: 700;
  color: black;
}

/* each element */
.interactives .element {
  background-color: #00dca5;
  border: none;
  border-radius: 36px;

  padding: 10px 20px;

  display: flex;
  align-items: center;

  user-select: none;
}

/* the location icon */
.interactives .element.location img {
  width: auto;
  margin-right: 5px;
  overflow: visible;
  overflow-clip-margin: 0px;
}

/* the share icon */
.interactives .share.element img {
  margin-right: 0px;
}

/* the location button */
```

```
.interactives .location {
  width: 100%;
  margin-right: 10px;
}
```

loadingScreen.css

> frontend/src/features/stylesheets/loadingScreen.css

Written by Eugene Au

```
.loading-screen {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.8);
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 9999;
}

/* .loading-text {
  color: #fff;
  font-size: 24px;
} */
```

polaroid.css

> frontend/src/features/stylesheets/polaroid.css

Written by Eugene Au

```
@import url("https://fonts.googleapis.com/css2?
family=Nanum+Pen+Script&display=swap");

/* the polaroid container */
.polaroid {
  background: white;
  margin-bottom: 10px;
  font-size: 20px;
}
```

```css
/* added padding */
.polaroid .padding {
  padding: 5% 3%;
}

/* the text */
.polaroid .caption {
  overflow-wrap: break-word;
  font-family: "Nanum Pen Script", cursive;
  font-weight: 400;
  font-style: normal;
  color: black;
  line-height: 1.2;
}

.polaroid img {
  max-height: 100%;
  width: auto;
}
```

postView.css

> frontend/src/features/stylesheets/postView.css

Written by Eugene Au

```css
.display.active {
  position: fixed; /* changed from absolute to fixed */
  box-sizing: border-box; /* fixed the height issue */
  top: 0; /* position from the top */
  left: 0; /* position from the left */

  /* occupy the whole screen */
  min-height: 100vh;
  min-width: 100vw;

  /* background color with opacity */
  background-color: rgba(25, 25, 25, 0.8);

  /* show the display */
  display: block;

  /* prevent the display from being clipped */
  overflow-y: auto;
  z-index: 10;

  /* center the wrapper */
  display: flex;
```

```
  justify-content: center;
  align-items: center;
}

/* animation for the post */
.display.active .post-wrapper .post {
  animation: fadeIn 0.3s ease-in-out;
}

@keyframes fadeIn {
  from {
    scale: 0.99;
    opacity: 0;
  }
  to {
    scale: 1;
    opacity: 1;
  }
}

.post-wrapper {
  /* space between viewport */
  width: 98%;
  /* don't let the post be too big */
  max-width: 500px;
  /* get all the space */
  height: 100%;
}

/* hide the display */
.display {
  display: none;
  position: absolute;
  max-width: 500px;
}

.post {
  /* avoid the post from being too big */
  max-width: 500px;

  max-height: 100vh;
  /* position the post in the center */
  position: absolute;
  transform: translateY(-50%);
  top: 50%;

  /* the width should be the same as the wrapper */
  width: inherit;
}
```

```
/* the date and time text */
.post .date-time {
  position: absolute;
  bottom: 14px;
  left: 14px;
  z-index: 2;
  font-size: 12px;
  font-weight: 700;
}
```

drawer-down.css

> frontend/src/styles/drawer-down.css

Written by Adam George

```
.drawer-wrapper {
  width: 100%;
  display: flex;
  justify-content: center;
}

/* Drawer styling */
.drawer-container {
  animation: slideOutTop 0.7s ease-out;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: flex-end;
  position: absolute;
  width: 100vw;
  background-color: white;
  height: 55vh;
  z-index: 1;

  /* make it so that the drawer dosnt get too wide */
  max-width: 500px;

  /* border radius on the bottom*/
  border: none;
  border-bottom-right-radius: 10px;
  border-bottom-left-radius: 10px;
}

.drawer-exit-top {
  animation: slideInTop 0.7s ease-in;
}
```

```css
.drawer-container .polaroid {
  box-shadow: 2px 2px 5px 0px rgba(0, 0, 0, 0.3);
}
.drawer-container button {
  font-family: Outfit, sans-serif;
  font-weight: 700;
  color: black;
  margin: 16px 0;
  height: 28px;
  width: 126px;
  border-radius: 50px;
  border: 0px;
  background-color: var(--primary);
  cursor: pointer;
}

.drawer-container button:disabled {
  color: #dfdfdf;
  background-color: #74d6be;
  cursor: not-allowed;
}

.drawer-container #handle {
  height: 5px;
  width: 43px;
  margin-bottom: 10px;
}

#texture {
  background: url(../assets/map/texture-small.jpeg);
  background-size: cover;
  height: 40vh;

  /* make it so that there's 20px margin on each side */
  width: calc(100% - 40px);
  border-bottom-left-radius: 10px;
  border-bottom-right-radius: 10px;
  overflow: auto;
  position: relative;
  /* account for the header */
  margin-top: 50px;

  /* center the polaroid */
  display: flex;
  align-items: center;
  justify-content: center;
}

#polariod-container {
  /* make it so that the width of the polaroid stays at 50% */
```

```
  width: 50%;
  max-width: 500px;
}
/* #texture::-webkit-scrollbar {
      width: 3px;
      height: 10px;
  }

  #texture::-webkit-scrollbar-track {
      background: #f1f1f1;
  }

  #texture::-webkit-scrollbar-thumb {
      background: #aaa;
      border-radius: 10px;
  }

  #texture::-webkit-scrollbar-thumb:hover {
      background: #999;
  } */

/* Animation css classes */
@keyframes fadeIn {
  from {
    transform: scale(0.5);
    opacity: 0;
  }
  to {
    opacity: 1;
    transform: scale(1);
  }
}

@keyframes slideInTop {
  from {
    transform: translateY(-100%);
  }
  to {
    transform: translateY(0%);
  }
}

@keyframes slideOutTop {
  from {
    transform: translateY(0%);
  }
  to {
    transform: translateY(-100%);
  }
}
```

Pages

capture.jsx

> frontend/src/pages/capture.jsx

Written by Eugene Au and Adam George

```
import { useEffect, useRef, useState } from "react";
import { useNavigate } from "react-router-dom";
import { usePositionStore, useGeoTagStore } from "../stores/geolocationStore";
import Send from "../assets/send.svg";
import Location from "../assets/location.svg";
import Reset from "../assets/reset.svg";
import axios from "axios";
import Cookies from "universal-cookie";
import Geolocation from "../features/Geolocation";
import LoadingScreen from "../features/loadingScreen";

import CheckLogin from "../features/CheckLogin";
import "./stylesheets/capture.css";

const cookies = new Cookies();
axios.defaults.withCredentials = true;

function Capture() {

  // check if the user is logged in
  useEffect(() => {
    async function check() {
      const res = await CheckLogin();
      capture();
    }
    check();
  }, []);

  // the navigate function
  const navigate = useNavigate();

  // Simple mobile detection
  // if (/Android|webOS|iPhone|iPad|iPod|BlackBerry|IEMobile|Opera
Mini/i.test(navigator.userAgent)) {
  //     // Trigger file input on mobile devices
  //     document.getElementById('fileInput').click();
  // } else {
  //     alert('Camera capture is optimized for mobile devices.');
```

```
  // }

  // the user's position
  const position = usePositionStore(state => state.position);
  const setPosition = usePositionStore(state => state.setPosition);
  const locationTag = useGeoTagStore(state => state.geoTag);
  const setLocationTag = useGeoTagStore(state => state.setGeoTag);

  const [lastPosition, setLastPosition] = useState({ lat: 0, lng: 0 });

  // the file input
  const inputRef = useRef(null);

  // the preview image
  const [previewImg, setPreviewImg] = useState("");
  const [file, setFile] = useState(null);

  // if the page is loading
  const [isLoading, setIsLoading] = useState(false);

  const [caption, setCaption] = useState("");

  // when the form is changed, set the state in post data
  const handleChange = (e) => {
    setCaption(e.target.value);
  };

  // get the user's position
  useEffect(() => {
    if (navigator.geolocation) {
      navigator.geolocation.watchPosition((position) => {
        setPosition(position.coords.latitude, position.coords.longitude);
      });
    }
  }, []);

  // set the location tag
  useEffect(() => {
    if (!(lastPosition.lat && lastPosition.lng) || Math.abs(lastPosition.lat -
position.lat) > 0.001 || Math.abs(lastPosition.lng - position.lng) > 0.001) {
      Geolocation(position.lat, position.lng, setLocationTag);
      setLastPosition({ lat: position.lat, lng: position.lng });
    }
  }, [position]);

  // on submit
  const handleSubmit = async (e) => {
    e.preventDefault();
    setIsLoading(true);
```

```
    try {
      const response = await axios.post(
        "http://127.0.0.1:8000/api/geolocations/",
        {
          latitude: position.lat,
          longitude: position.lng,
          location: locationTag,
        }
      );
      const geolocID = response.data.id;
      const formData = new FormData();
      // Ensure 'fileInput' is the correct ID for your file input field
      const imageFile = document.getElementById("fileInput").files[0];
      if (imageFile) {
        formData.append("image", imageFile);
      } else {
        alert("No image file selected");
        setIsLoading(false);
        return; // Exit the function if no file is selected
      }

      // Append other postData fields to formData
      // formData.append("userid", 1);
      formData.append("caption", caption);
      formData.append("geolocID", geolocID); // Append geolocID as a number

      try {
        // Update the API URL as per your configuration
        const response = await axios.post(
          "http://127.0.0.1:8000/api/createPost/",
          formData,
          {
            headers: {
              "Content-Type": "multipart/form-data",
              "Authorization": `Token ${cookies.get('token')}`, // Assuming
postData.username is the token
            },
          }
        );

        // Redirect to the home page after successful post creation
        navigate("/");
      } catch (error) {
        console.error("Error occurred:", error);
        if (error.response) {
          console.log("Response data:", error.response.data);
          console.log("Response status:", error.response.status);
          console.log("Response headers:", error.response.headers);

          alert("An error occurred while creating the post");
```

```
      }
    }
  } catch (error) {
    console.error("Error occurred:", error);
    if (error.response) {
      console.log("Response data:", error.response.data);
      console.log("Response status:", error.response.status);
      console.log("Response headers:", error.response.headers);
    }

    alert("An error occurred while creating the post");
  }

  setIsLoading(false);
};

// triggered when the user clicks the capture button
const capture = () => {
  // Trigger file input
  inputRef.current.click();
};

// handle the capture
const handleCapture = (e) => {
  setPreviewImg(URL.createObjectURL(e.target.files[0]));
  console.log(previewImg);

  if (file) {
    setPreviewImg(URL.createObjectURL(file));
    setFile(file); // store the file object
  }
};

return (
  <>

    <LoadingScreen active={isLoading} />
    <div id="capturePage" className="page active">
      {/* the invisible file input element */}
      <input
        type="file"
        accept="image/*"
        capture="environment"
        id="fileInput"
        ref={inputRef}
        onChange={handleCapture}
        style={{ display: "none" }}
      />
      <div id="display">
        <div id="preview-wrapper">
```

```
<div id="preview">
  <div id="spacer">
    <div id="contents">
      {/* the preview image */}
      {previewImg ? (
        <img
          src={previewImg}
          alt="Preview Image"
          style={{ maxWidth: "100%", height: "auto" }}
        />
      ) : (
        <div
          id="previewPlaceholder"
          onClick={() => {
            capture();
          }}
        >
          <p>Tap to take a picture</p>
        </div>
      )}

      {/* the form 70px margin for the footer*/}
      <form onSubmit={handleSubmit}>
        {/* the caption */}
        <div id="form">
          <input
            name="caption"
            type="caption"
            placeholder="post"
            onChange={handleChange}
          />
        </div>

        {/* the buttons on the bottom */}
        <div id="previewButtons">
          {/* the retake button */}
          <div className="retake element">
            <img
              src={Reset}
              width={"15px"}
              height={"15px"}
              onClick={() => {
                capture();
              }}
            />
          </div>

          {/* the location button */}
          <div className="location element">
            <img src={Location} />
```

```
                        {locationTag}
                      </div>

                      {/* the submit button */}
                      <button
                        className="share element"
                        // type="submit"
                        style={{ height: "100%" }}
                      >
                        <img src={Send} />
                      </button>
                    </div>
                  </form>
                </div>
              </div>
            </div>
          </div>
        </div>
      </>
    );
}

export default Capture;
```

feed.jsx

> frontend/src/pages/feed.jsx

Written by Eugene Au

```
import React from "react";
import "./stylesheets/feed.css";
import { useState, useEffect } from "react";
import PostView from "../features/PostView";
import Polaroid from "../features/polaroid";
import { set } from "date-fns";
import axios from "axios";
import Cookies from "universal-cookie";
import CheckLogin from "../features/CheckLogin";


const cookies = new Cookies();

function FeedPage() {

  // check if the user is logged in
  CheckLogin();
```

```
  const [activePost, setActive] = useState({});
  const [columns, setColumns] = useState([]);

  // get all the post from database
  const getPosts = async () => {

    const token = cookies.get('token');

    try {
      // Update the API URL as per your configuration
      const response = await axios.post(
        "http://127.0.0.1:8000/api/collectedPosts/",
        {},
        {
          headers: {
            "Content-Type": "multipart/form-data",
            "Authorization": `Token ${token}`, // Assuming postData.username is the
token
          },
        }
      );
      return response.data;
    } catch (error) {
      console.error("Error occurred:", error);
      if (error.response) {
        console.log("Response data:", error.response.data);
        console.log("Response status:", error.response.status);
        console.log("Response headers:", error.response.headers);
      }
    }
  };

  useEffect(() => {
    // Function to load an image and update its height in the state
    function getImageHeight(url) {
      // wait for the image to load
      return new Promise((resolve, reject) => {
        // create a new image
        const img = new Image();

        // when the image loads, resolve with the height
        img.onload = () => {
          resolve(img.height);
        };

        // if there is an error, reject with the error
        img.onerror = reject;
        img.src = url;
      });
```

```
    }

    // Distribute the images into two columns based on which column is shorter
    // Also randomize the rotation of the images
    const processImages = async (e) => {

      const postList = await getPosts();

      let heightDifference = 0;
      const rightPosts = [];
      const leftPosts = [];

      for (let i = 0; i < postList.length; i++) {

        postList[i]["image"] = "http://127.0.0.1:8000/" + postList[i]["image"]
        const image = postList[i]["image"];

        // add rotation
        postList[i]["rotation"] = -2 + Math.random() * (2 + 2);

        const imageHeight = await getImageHeight(image);

        // if the right column is shorter, add the image to the right column
        if (heightDifference < 0) {
          rightPosts[i] = postList[i];
          heightDifference += imageHeight + 10; //10 for the margin
        } else {
          leftPosts[i] = postList[i];
          heightDifference -= imageHeight + 10; //10 for the margin
        }
        setColumns([leftPosts, rightPosts]);
      }

    };

    processImages();

  }, []);

  return (
    <>
      {/* the absolute position post view */}
      <PostView
        isActive={Object.keys(activePost).length !== 0}
        image={activePost["image"]}
        leaveFunction={() => {
          setActive({});
        }}
        caption={activePost["caption"]}
        location={"Forum"}
```

```
      />

      {/* the feed */}
      <div id="feed">
        <div id="daily-feed">
          <div id="grid-wrapper">
            {/* map each columns */}
            {columns.map((column, index) => (
              <div key={index} className={"image-grid " + index}>
                {/* map each posts in the column */}
                {column.map((post) => (
                  <div className={post["id"]} key={post["id"]}>
                    <Polaroid
                      src={post["image"]}
                      func={() => {
                        setActive(post);
                      }}
                      caption={post["caption"]}
                      rotation={post["rotation"]}
                    />
                  </div>
                ))}
              </div>
            ))}
          </div>
        </div>
      </div>
    </>
  );
}

export default FeedPage;
```

login.jsx

> frontend/src/pages/login.jsx

Written by Eugene Au

```
import "./stylesheets/login.css";
import { Link, useNavigate } from "react-router-dom";
import axios from "axios";
import { useState } from "react";
import Cookies from "universal-cookie";

const cookies = new Cookies();

function LoginPage() {
```

```
const navigate = useNavigate();

const [errors, setErrors] = useState({
  username: "",
  password: "",
});

const [userData, setUserData] = useState({
  username: "",
  password: "",
});

const handleChange = (e) => {
  setUserData({
    ...userData,
    [e.target.name]: e.target.value,
  });
};

const hansleSubmit = async (e) => {
  e.preventDefault();
  console.log(userData);
  try {
    const response = await axios.post(
      "http://127.0.0.1:8000/api/login/",
      userData
    );
    cookies.set("token", response.data.token, { path: "/" });
    navigate("/");
    return
  }
  catch (error) {
    // Display the error message from the server
    if (error.response) {
      if (error.response.data.username) {
        setErrors({
          ...errors,
          username: error.response.data.username,
        });
        return
      } else {
        console.log(error);
        alert("An error occured. Please try again later.");
      }
    }
  }
  console.log(errors);
  alert("An error occured. Please try again later.");
}
```

```
  return (
    <div id="display">
      <div id="login-wrapper">
        <div id="login">
          {/* a spacer for the hader and footer  */}
          <div id="spacer">
            <div id="title">Login</div>
            <form onSubmit={hansleSubmit}>
              <div id="form">
                <div className="field">
                  <input
                    name="username"
                    className="text"
                    type="text"
                    placeholder="Username"
                    onChange={handleChange}
                  />
                  <label className="error">{errors.username}</label>
                </div>
                <div className="field">
                  <input
                    name="password"
                    className="text"
                    type="password"
                    placeholder="Password"
                    onChange={handleChange}
                  />
                  <label className="error">{errors.username}</label>
                </div>
                <button>Login</button>
                <div id="message">
                  Not a member?{" "}
                  <Link to={"/register"}>
                    Sign Up
                  </Link>
                </div>
              </div>
            </form>
          </div>
        </div>
      </div >
    </div >
  );
}

export default LoginPage;
```

map.jsx

frontend/src/pages/map.jsx

Written by Adam George

```
import {
  APIProvider,
  Map,
  AdvancedMarker,
  Marker,
  Pin,
  useMap,
} from "@vis.gl/react-google-maps";
import { useState, useEffect, useMemo } from "react";
import { differenceInDays, format, set } from "date-fns";
import { PathLayer } from "@deck.gl/layers";
import { GoogleMapsOverlay } from "@deck.gl/google-maps";
import { usePositionStore, useGeoTagStore } from "../stores/geolocationStore";
import { usePinStore, useCollectedPinStore } from "../stores/pinStore";
import "./stylesheets/map.css";
import MoodPrompt from "../features/MoodPrompt";
import DrawerDown from "../features/DrawerDown";
import Location from "../assets/location.svg";
import axios from "axios";
import InitMap from "../features/InitMap";
import Geolocation from "../features/Geolocation";
import Cookies from "universal-cookie";
import PostView from "../features/PostView";
import CheckLogin from "../features/CheckLogin";

const cookies = new Cookies();

// Debugging options
const seeAllPins = true;
const spoofLocation = false;

// Overlay constructor component (from deck.gl documentation)
export const DeckGlOverlay = ({ layers }) => {
  const deck = useMemo(() => new GoogleMapsOverlay({ interleaved: true }), []);

  const map = useMap();
  useEffect(() => deck.setMap(map), [map]);
  useEffect(() => deck.setProps({ layers }), [layers]);

  return null;
};

// get all the post from database
const getPosts = async () => {
  try {
```

```
    const response = await axios.get(
      "http://127.0.0.1:8000/api/getRecentPosts/"
    );
    return response.data;
  } catch (error) {
    console.error(error);
  }
};

// Get collected posts from the database
const getCollectedPosts = async (token) => {
  try {
    const response = await axios.post(
      "http://127.0.0.1:8000/api/collectedPosts/",
      {},
      {
        headers: {
          "Content-Type": "multipart/form-data",
          "Authorization": `Token ${token}`,
        },
      }
    );
    return response.data;
  } catch (error) {
    console.error(error);
  }
};

function MapPage() {

  // check if the user is logged in
  CheckLogin()

  // State for active post in the view
  const [activePost, setActive] = useState({});

  const [loading, setLoading] = useState(true);
  const [progress, setProgress] = useState(20);

  // State for mood prompt
  const [showMoodPrompt, setShowMoodPrompt] = useState(false);
  const [mood, setMood] = useState("unselected");

  // State for drawer
  const [drawerTopVisible, setDrawerTopVisible] = useState(false);
  const [drawerPost, setDrawerPost] = useState(null);

  // State for walking and tracking path coordinates and map position
  const [path, setPath] = useState([]);
  const [walking, setWalking] = useState(false);
```

```
  const [watchId, setWatchId] = useState(null);
  const [lastPosition, setLastPosition] = useState({ lat: undefined, lng: undefined
});

  // State for form data ( adding posts to collection )
  const [form, setForm] = useState({ "postid": 0 })

  // Global state for current position, location tag and pins
  const position = usePositionStore(state => state.position);
  const setPosition = usePositionStore(state => state.setPosition);
  const locationTag = useGeoTagStore(state => state.geoTag);
  const setLocationTag = useGeoTagStore(state => state.setGeoTag);
  const pins = usePinStore(state => state.pins);
  const setPins = usePinStore(state => state.setPins);

  // Global state for collected pins
  const collectedPins = useCollectedPinStore(state => state.pinIds);
  const setCollectedPins = useCollectedPinStore(state => state.setPinIds);
  const addCollectedPin = useCollectedPinStore(state => state.addPinId);

  const token = cookies.get('token');

  // Sample data for path layer (to be replaced as well)
  const path2 = [
    {
      path: [
        [-3.532736, 50.733763],
        [-3.532653, 50.733856],
        [-3.532582, 50.734025],
        [-3.532538, 50.734098],
        [-3.532489, 50.734238],
        [-3.532412, 50.734407],
        [-3.532396, 50.734417],
        [-3.532224, 50.734756],
        [-3.532171, 50.734879],
        [-3.531977, 50.735076],
        [-3.531859, 50.735137],
        [-3.531945, 50.735259],
        [-3.532063, 50.735374],
        [-3.532138, 50.735442],
        [-3.532407, 50.735619],
        [-3.532825, 50.735802],
        [-3.532825, 50.735802],
        [-3.533136, 50.735856],
        [-3.533415, 50.73585],
      ],
    },
  ];

  // Layer constructor for path layer (from deck.gl documentation)
```

```
  const layer = new PathLayer({
    id: "path-layer",
    data: path,
    getPath: (d) => d.path,
    getColor: [73, 146, 255],
    getWidth: 7,
    widthMinPixels: 2,
  });

  useEffect(() => {
    if (progress >= 100) {
      setTimeout(() => {
        setLoading(false);
      }, 2550);
    }
  }, [progress]);

  useEffect(() => {
    new Promise((resolve, reject) => {
      // Get the user's current position
      if (navigator.geolocation) {
        navigator.geolocation.watchPosition((position) => {
          setPosition(position.coords.latitude, position.coords.longitude);
        });
      }
      setProgress(oldProgress => oldProgress + 30);
      resolve();
    })
      .then(() => {
        // Checks if mood has been set before, if not call mood prompt;
        if (sessionStorage.mood === undefined) {
          setShowMoodPrompt(true);
        } else {
          setMood(sessionStorage.mood);
        }
        setProgress(oldProgress => oldProgress + 30);
      })
      .then(() => {
        cookies.get('token');
        getCollectedPosts(token).then((data) => data.map((post) =>
addCollectedPin(post.id)));
        getPosts().then((data) => {
          setPins(data);
          setProgress(oldProgress => oldProgress + 20);
        });
      });
  }, []);

  // useEffect(() => {
  //   if (navigator.geolocation && walking) {
```

```
  //     const watchId = navigator.geolocation.watchPosition((position) => {
  //       setPosition(position.coords.latitude, position.coords.longitude);
  //       setPath((currentPath) => [...currentPath, [position.coords.longitude,
position.coords.latitude]]);
  //       setWatchId(watchId);
  //     });
  //   } else if (!walking) {
  //     navigator.geolocation.clearWatch(watchId);
  //   }
  // }, [walking]);

  useEffect(() => {
    if (!(lastPosition.lat && lastPosition.lng) || Math.abs(lastPosition.lat -
position.lat) > 0.001 || Math.abs(lastPosition.lng - position.lng) > 0.001) {
      Geolocation(position.lat, position.lng, setLocationTag);
      setLastPosition({ lat: position.lat, lng: position.lng });
    }
  }, [position]);

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      // Update the API URL as per your configuration
      const response = await axios.post(
        "http://127.0.0.1:8000/api/collectPost/",
        form,
        {
          headers: {
            "Content-Type": "multipart/form-data",
            "Authorization": `Token ${token}`, // Assuming postData.username is the
token
          },
        }
      );
      addCollectedPin(form.postid);
      console.log(response.data);
    } catch (error) {
      console.error("Error occurred:", error);
      if (error.response) {
        console.log("Response data:", error.response.data);
        console.log("Response status:", error.response.status);
        console.log("Response headers:", error.response.headers);
      }
    }
  };

  // Filter pins based on radial distance calculated using the Haversine formula
  const filterPins = (lat, lng) => {
    const radius = 0.0005; // Radius of tolerance (about 35m from the position)
```

```
  const closePins = pins.filter((pin) => {
    const dLat = deg2rad(pin.position.lat - lat);
    const dLng = deg2rad(pin.position.lng - lng);
    const a =
      Math.sin(dLat / 2) * Math.sin(dLat / 2) +
      Math.cos(deg2rad(lat)) * Math.cos(deg2rad(pin.position.lat)) *
      Math.sin(dLng / 2) * Math.sin(dLng / 2)
      ;
    const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
    const distance = c * 6371.1; // Distance of the Earth's radius (km)

    return distance < radius;
  });
  return seeAllPins ? pins : closePins;
}

// Converts degrees to radians
function deg2rad(deg) {
  return deg * (Math.PI / 180)
}

const handleOpen = (e, id) => {
  // prevent the user from clicking into the outsite area and the map icon
  e.domEvent.preventDefault();

  setPins(
    pins.map((pin) => {
      if (pin.id === id) {
        setForm({ "postid": id })
        setDrawerTopVisible(true);
        setDrawerPost(pin);
        return { ...pin, open: !pin.open };
      } else {
        return { ...pin, open: false };
      }
    })
  );
};

const handleMoodPrompt = (mood) => {
  setMood(mood);
  sessionStorage.mood = mood;
  setShowMoodPrompt(false);
};

return (
  <>
    {/* the absolute position post view */}
    <PostView
      isActive={Object.keys(activePost).length !== 0}
```

```
        image={"http://127.0.0.1:8000/" + activePost['image']}
        leaveFunction={() => {
          setActive({});
        }}
        caption={activePost["caption"]}
        // perform a null check or ensure that activePost["position"]["location"]
exists before accessing its location property.
        location={activePost["position"] && activePost["position"]["location"]}
      />

      {loading && <InitMap progress={progress} />}
      <DrawerDown
        id={form.postid}
        image={"http://127.0.0.1:8000/" + drawerPost?.image}
        caption={drawerPost?.caption}
        drawerVisible={drawerTopVisible}
        setDrawerVisible={setDrawerTopVisible}
        handleSubmit={handleSubmit}
        handleClickPolaroid={() => setActive(pins.find((pin) => pin.id ===
form.postid))}
      />
      {(position.lat && position.lng) && <APIProvider apiKey=
{import.meta.env.VITE_GOOGLE_MAPS_API_KEY}>
        <div
          className={
            showMoodPrompt ? "mapContainer background-blur" : "mapContainer"
          }
        >
          <Map
            id="map"
            defaultCenter={position}
            defaultZoom={17}
            gestureHandling={"greedy"}
            disableDefaultUI={true}
            mapId={import.meta.env.VITE_GOOGLE_MAPS_MAP_ID}
          >
            <DeckGlOverlay layers={layer} />
            <AdvancedMarker key="current-position" position={position}>
              <div
                style={{
                  width: 16,
                  height: 16,
                  position: "absolute",
                  top: 0,
                  left: 0,
                  background: "#4185f5",
                  border: "2px solid #ffffff",
                  borderRadius: "50%",
                  transform: "translate(-50%, -50%)",
                }}
```

```
              ></div>
            </AdvancedMarker>
            {filterPins(position.lat, position.lng).map((pin) => {
              const color = `var(--${mood})`;
              return (
                <>
                  <AdvancedMarker
                    key={pin.id}
                    position={pin.position}
                    onClick={(e) => handleOpen(e, pin.id)}
                  >
                    <Pin
                      background={color}
                      borderColor={color}
                      glyphColor="white"
                      scale={0.8}
                    ></Pin>
                  </AdvancedMarker>
                </>
              );
            })}
          </Map>
        </div>
      </APIProvider>}
      {showMoodPrompt && <MoodPrompt onClickFunction={handleMoodPrompt} />}
      <div className="bottom-prompt">
        <div className="bottom-prompt-wrapper">
          <img src={Location} />{locationTag}
        </div>
      </div>
    </>
  );
}

export default MapPage;
```

pageNoFound.jsx

> frontend/src/pages/pageNoFound.jsx

Written by Eugene Au and Adam George

```
import "./stylesheets/pageNoFound.css";
import buddy from "../assets/404buddy.jpg"

function PageNoFound() {
  return (
    <div id="display-not-found">
```

```
      <img id="not-found-img" src={buddy}></img>
      <div id="status-not-found">404</div>
      <p id="description-not-found">Uh oh! We can't find the page you were looking
for</p>
    </div>
  );
}

export default PageNoFound;
```

profilepage.jsx

> frontend/src/pages/profilepage.jsx

Written by Eugene Au

```
import "./stylesheets/profilepage.css";

import userIcon from "../assets/header/user-icon.jpg";

function ProfilePage() {
  return (
    <div id="display">
      <div id="profile-wrapper">
        <div id="profile">
          <h1>Profile</h1>
          <div id="user-icon">
            <img src={userIcon} alt="user icon" width={"100%"} />
          </div>
        </div>
      </div>
    </div>
  );
}

export default ProfilePage;
```

register.jsx

> frontend/src/pages/register.jsx

Written by Eugene Au

```
import "./stylesheets/register.css";
import { useState } from "react";
```

```
import { useNavigate } from "react-router-dom";
import axios from "axios";

import Cookies from "universal-cookie";

import LoadingScreen from "../features/loadingScreen";

const cookies = new Cookies();

function RegisterPage() {
  const navigate = useNavigate();

  const [isLoading, setIsLoading] = useState(false);

  const [userData, setUserData] = useState({
    username: "",
    email: "",
    password: "",
    confirmPassword: "",
    checkbox: false,
  });

  const [errors, setErrors] = useState({
    username: "",
    email: "",
    password: "",
  });

  const [isChecked, setIsChecked] = useState(false);

  const handleCheckBoxChange = () => {
    setUserData({
      ...userData,
      checkbox: !isChecked,
    });
    setIsChecked(!isChecked);
  };

  const handleChange = (e) => {
    setUserData({
      ...userData,
      [e.target.name]: e.target.value,
    });
  };

  const handleSubmit = async (e) => {
    // prevent the default form submission
    e.preventDefault();
    setIsLoading(true);
    // if the password and confirm password do not match, display an error message
```

```
    if (userData.password !== userData.confirmPassword) {
      setErrors({
        ...errors,
        confirmPassword: "Passwords do not match",
      });
      setIsLoading(false);
      return;
    }
    // if the checkbox is not checked, display an error message
    if (!userData.checkbox) {
      setErrors({
        ...errors,
        checkbox: "Please agree to the terms and conditions",
      });
      setIsLoading(false);
      return;
    }

    // if the password dose not meet the requirements, display an error message
    if (userData.password.length < 8) {
      setErrors({
        ...errors,
        password: "Password must be at least 8 characters long",
      });
      setIsLoading(false);
      return;
    }

    // remove all the error messages
    setErrors({
      username: "",
      email: "",
      password: "",
      confirmPassword: "",
      checkbox: "",
    });

    // submit the form data to the server
    try {
      const response = await axios.post(
        "http://127.0.0.1:8000/api/register/",
        userData
      );
      cookies.set("token", response.data.token, { path: "/" });
      navigate("/");
    } catch (error) {
      if (error.response) {
        // The server responded with a status code outside the range of 2xx
        console.error("Error data:", error.response.data);
        // Display the error message from the server
```

```
        if (error.response.data.username) {
          setErrors({
            ...errors,
            username: error.response.data.username,
          });
        }
        if (error.response.data.email) {
          setErrors({
            ...errors,
            email: error.response.data.email,
          });
        }
      } else {
        // The request was made but no response was received or other errors
occurred
        console.error("Error:", error.message);
      }

      alert("An error occurred.")
      // remove the loading screen
      setIsLoading(false);
      return;
      // Handle error (e.g., show error message)
    }
  };

  return (
    <>
      <LoadingScreen active={isLoading} />
      <div id="display">
        <div id="register-wrapper">
          <div id="register">
            <div id="spacer">
              <div id="title">Registration</div>
              <div id="form">
                <form onSubmit={handleSubmit}>
                  <div className="field">
                    <input
                      type="text"
                      name="username"
                      value={userData.username}
                      onChange={handleChange}
                      placeholder="Username"
                      required
                      className="text input"
                    />
                    <label className="error">{errors.username}</label>
                  </div>
                  <div className="field">
                    <input
```

```
            type="email"
            name="email"
            value={userData.email}
            onChange={handleChange}
            placeholder="Email"
            required
            className="text input"
          />
          <label className="error">{errors.email}</label>
        </div>
        <div className="field">
          <input
            type="password"
            name="password"
            value={userData.password}
            onChange={handleChange}
            placeholder="Password"
            required
            className="text input"
          />
          <label className="error">{errors.password}</label>
        </div>
        <div className="field">
          <input
            className="text input"
            type="password"
            placeholder="Confirm Password"
            name="confirmPassword"
            onChange={handleChange}
            required
          />
          <label className="error">{errors.confirmPassword}</label>
        </div>
        <div className="field">
          <label className="checkbox">
            <input
              type="checkbox"
              onChange={handleCheckBoxChange}
              checked={isChecked}
            />
            <span className="checkmark"></span>I agree to Postpal's
            Terms & Conditions and Privacy Policy
            <br />
          </label>
          <label className="error">{errors.checkbox}</label>
        </div>
        <button type="submit">Register</button>
      </form>
    </div>
  </div>
```

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

export default RegisterPage;
```

test.jsx

> frontend/src/pages/test.jsx

Written by Eugene Au

```
import React, { Component } from "react";
import axios from "axios";

import Cookies from "universal-cookie";
import './stylesheets/test.css'
import { useState } from "react";
import CheckLogin from "../features/CheckLogin";

const cookies = new Cookies();

function test() {

  // CheckLogin();

  const token = cookies.get('token');

  console.log(token);

  const [form, setForm] = useState({
    "postid": 2
  })

  const handleSubmit = async (e) => {

    e.preventDefault();
    try {
      // Update the API URL as per your configuration
      const response = await axios.get(
        "http://127.0.0.1:8000/api/getUser/"
        ,
        {
          headers: {
            "Content-Type": "multipart/form-data",
```

```
            "Authorization": `Token ${cookies.get('token')}`, // Assuming
postData.username is the token
          },
        }
      );
      console.log(response.data);
    } catch (error) {
      console.error("Error occurred:", error);
      if (error.response) {
        console.log("Response data:", error.response.data);
        console.log("Response status:", error.response.status);
        console.log("Response headers:", error.response.headers);
      }
    }
  }

  return (
    <div className="test">
      <form onSubmit={handleSubmit}>
        <button type="submit">submit</button>
      </form>
    </div>
  );
}

export default test;
```

capture.css

> frontend/src/pages/stylesheets/capture.css

Written by Eugene Au

```
#display {
  position: fixed; /* changed from absolute to fixed */
  box-sizing: border-box; /* fixed the height issue */
  top: 0; /* position from the top */
  left: 0; /* position from the left */

  /* occupy the whole screen */
  min-height: 100vh;
  min-width: 100vw;

  /* show the display */
  display: block;

  /* prevent the display from being clipped */
  overflow-y: auto;
```

```
  /* center the wrapper */
  display: flex;
  justify-content: center;
  align-items: center;
}

#preview-wrapper {
  /* 20px padding around the side */
  width: calc(100% - 40px);
}

#preview {
  position: absolute;
  width: inherit;
  max-height: 100vh;
  transform: translateY(-50%);
  top: 50%;
}

#spacer {
  padding: 70px 0;
  margin-bottom: 120px;
  overflow: scroll;
  max-width: 500px;
  min-width: 200px;
  margin-left: auto;
  margin-right: auto;
}

#preview #contents {
  width: 100%;
  max-width: 500px;
}

#preview img {
  max-width: 500px;
  width: 100%;
  height: 100%;
  border: none;
  border-radius: 10px;
}

#previewButtons {
  height: 35px;
  width: 100%;
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-top: 10px;
```

```
  font-size: 12px;
  font-weight: 700;
  color: black;
  overflow: visible;
}

#previewButtons .element {
  background-color: #00dca5;
  border: none;
  border-radius: 36px;
  padding: 10px 20px;
  display: flex;
  align-items: center;
  height: 15px;
}

#previewButtons .like {
  justify-content: center;
}

#previewButtons .element img {
  width: 15px;
  height: 15px;
  margin-right: 5px;
}

#previewButtons .share.element img {
  margin-right: 0px;
}

#previewButtons .share.element {
  background-color: whitesmoke;
}

#previewButtons .location {
  width: 100%;
  margin: 0 10px;
}

#previewPlaceholder {
  display: flex;
  justify-content: center;
  align-items: center;
  aspect-ratio: 16/9;
  text-align: center;
  border: black 1px dashed;
  border-radius: 10px;
  color: black;
}
```

```
#preview #form {
  width: 100%;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  margin-top: 20px;
}

#preview #form input {
  /* make the input field take up the whole width */
  width: 100%;

  /* styles */
  font-size: 16px;
  border: none;
  border-radius: 100px;
  padding: 13px 21px;
  font-weight: 700;
  color: black;
  background-color: #e7e7e7;
  margin-bottom: 20px;

  /* make the form start from the left */
  box-sizing: border-box;

  /* Remove the red border when selected */
  outline: none;
}
```

feed.css

> frontend/src/pages/stylesheets/feed.css

Written by Eugene Au and Adam George

```
#feed {
  padding: 80px 20px;
  max-width: 500px;
  margin-left: auto;
  margin-right: auto;
}

#top {
  margin-bottom: 25px;

  position: relative;
  display: inline-block;
```

```
  width: 100%;
  height: auto;
}

#top img {
  width: 100%;
  aspect-ratio: 5/4;
  object-fit: cover;
  border-radius: inherit;
  background-color: wheat;
}

#top #image-wrapper {
  position: relative;
  width: 100%;
  display: flex;
  border: none;
}

#daily-feed {
  color: black;
  font-size: 20px;
  font-weight: 600;
}

#daily-feed img {
  border: none;
}

#daily-feed #title {
  width: fit-content;
  line-height: 10px;
}

.image-grid {
  width: 49%; /* Make images fill their cell */
  row-gap: 10px;
  column-gap: 10px;
}

.image-grid .daily-feed-post {
  width: 100%; /* Make images fill their cell */
  height: auto; /* Maintain aspect ratio */
  object-fit: cover; /* Adjust this as needed */
}

#grid-wrapper {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
```

```
}

#feed {
  background-image: url("../../assets/feed/oak-wood-texture-design-background.jpg");
  background-size: 100% auto;
  background-repeat: repeat;
  min-height: 100vh;
}

#feed .polaroid {
  animation: polaroidFadeIn 0.8s ease-out;
}

/* Animation css classes */
@keyframes polaroidFadeIn {
  from {
    margin-top: 70%;
    opacity: 0.5;
    scale: 0.7;
    transform: rotateY(30deg);
  }
  to {
    opacity: 1;
    margin-top: 0%;
    scale: 1;
    transform: rotateY(0deg);
  }
}
```

login.css

> frontend/src/pages/stylesheets/login.css

Written by Eugene Au

```
/* center the content */
#display {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  width: 100vw;
}

/* adding padding */
#login-wrapper {
  width: calc(100% - 40px);
}
```

```css
/* the form */
#login {
  position: absolute;
  width: inherit;

  /* center the content from top to bottom */
  max-height: 100vh;
  transform: translateY(-50%);
  top: 50%;
}

/* the spacer for header and footer */
#spacer {
  padding: 70px 0;
  margin-bottom: 120px;
  overflow: scroll;

  max-width: 500px;
  min-width: 200px;

  margin-left: auto;
  margin-right: auto;
}

/* The title of the page */
#login #title {
  font-size: 50px;
  font-weight: 700;
  color: black;
  display: flex;
  justify-content: center;
  align-items: center;

  margin-bottom: 30px;
}

/* text input field's spacing */
#login #form .field {
  margin-bottom: 30px;
  width: 100%;
}

/* the forms and button  */
#login #form {
  /* center the content from top to bottom */
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
```

```
  width: 100%;
}

/* each input field */
#login #form input.text {
  /* make the input field take up the whole width */
  width: 100%;

  /* styles */
  font-size: 16px;
  border: none;
  border-radius: 100px;
  padding: 13px 21px;
  font-family: Outfit, sans-serif;
  font-weight: 600;
  color: black;
  background-color: #e7e7e7;

  /* make the form start from the left */
  box-sizing: border-box;

  /* Remove the red border when selected */
  outline: none;
}

#login #form button {
  width: 100%;

  /* styles */
  font-size: 16px;
  font-weight: 700;
  border: none;
  border-radius: 100px;
  padding: 10px;
  color: black;
  background-color: #00dca5;
  margin-bottom: 45px;
  font-family: "Outfit";
}

#form #message {
  color: black;
  margin-bottom: 20px;
  font-size: 16px;
  font-weight: 700;
}

#login #message {
  width: 100%;
```

```
  margin-bottom: 70px;
}
```

map.css

> frontend/src/pages/stylesheets/map.css

Written by Adam George

```
.mapContainer {
  height: 94vh;
  width: 100vw;
}

/* Override Google Maps default styling (move the close button further in window) */
button.gm-ui-hover-effect {
  top: -2px !important;
  right: -2px !important;
}

/* gm-style css classes are used to style Google Maps information windows (when a
user clicks on a marker) */
div.gm-style-iw.gm-style-iw-c {
  padding: 7px;
  width: calc(max-content - 30px);
  border-radius: 12px;
}

div.poi-info-window.gm-style,
.poi-info-window div,
.poi-info-window a {
  background: white;
  font-family: Outfit, sans-serif;
  color: black;
  padding: 1px;
}

div.gm-style-iw-d {
  margin-right: -10px;
  margin-bottom: -8px;
  background: white;
}

.gm-style .gm-style-iw-d::-webkit-scrollbar-track,
.gm-style .gm-style-iw-d::-webkit-scrollbar-track-piece,
.gm-style .gm-style-iw-c,
.gm-style .gm-style-iw-t::after,
.gm-style .gm-style-iw-tc::after {
```

```
  background: white;
}

/* Styling classes for mood prompt */
.mood-prompt {
  position: absolute;
  top: 0;
  left: 0;
  width: 100vw;
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
}

#mood-prompt-card {
  animation: fadeIn 1s ease-in;
  position: relative;
  max-width: 282px;
  width: 75vw;
  height: 301px;
  display: flex;
  flex-direction: column;
  align-items: center;
  border-radius: 10px;
  gap: 11px;
  background-color: white;
}

#mood-prompt-card img {
  position: absolute;
  top: 10px;
  right: 12px;
}

#mood-prompt-card p {
  margin-top: 24px;
  font-weight: 600;
  color: black;
}

#mood-prompt-card button {
  font-family: Outfit, sans-serif;
  font-weight: 600;
  height: 27px;
  width: 159px;
  border-radius: 50px;
  border: 0px;
  cursor: pointer;
  transition: 1s ease-out;
```

```
}

#mood-prompt-card button:hover,
#mood-prompt-card button:active {
  transform: scale(0.95);
  transition: 1s ease;
  filter: brightness(85%);
}

#mood-prompt-card button:active {
  transform: scale(0.95);
  transition: 1s ease;
}

#mood-prompt-card .urgent-help {
  color: gray;
  font-size: 12px;
  margin-top: 11px;
  margin-bottom: 21px;
}

.background-blur {
  filter: blur(2px);
}

.bottom-prompt {
  position: absolute;
  bottom: 88px;
  width: 100vw;
  display: flex;
  justify-content: center;
}

.bottom-prompt-wrapper {
  color: black;
  font-size: 14px;
  font-weight: 600;
  border-radius: 50px;
  background-color: var(--primary);
  padding: 5px 20px;
  display: flex;
  align-items: center;
  gap: 7px;
}
```

pageNoFound.css

> frontend/src/pages/stylesheets/pageNoFound.css

Written by Eugene Au and Adam George

```
.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 20px;
}

#status-not-found {
  color: #333;
  font-size: 40px;
  font-weight: bold;
}

#description-not-found {
  margin-top: -2px;
  color: #666;
  font-size: 20px;
  font-weight: 400;
  text-align: center;
  padding: 0 40px;
}

.error-message {
  color: red;
  font-weight: bold;
}

#display-not-found {
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
}

#not-found-img {
  width: 200px;
  height: 200px;
  margin-bottom: 20px;
}
```

profilepage.css

> frontend/src/pages/stylesheets/profilepage.css

Written by Eugene Au

```css
/* center the content */
#display {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  width: 100vw;
  color: #000;
}

/* adding padding */
#profile-wrapper {
  width: calc(100% - 40px);
}

/* the form */
#profile {
  position: absolute;
  width: inherit;

  /* center the content from top to bottom */
  max-height: 100vh;
  transform: translateY(-50%);
  top: 50%;
}

/* the spacer for header and footer */
#spacer {
  padding: 70px 0;
  margin-bottom: 120px;
  overflow: scroll;

  max-width: 500px;
  min-width: 200px;

  margin-left: auto;
  margin-right: auto;
}

#display h1 {
  text-align: center;
  font-size: 20px;
}

#profile img {
  width: 120px;
  height: 120px;
}
```

```
#profile #user-icon {
  width: 120px;
  height: 120px;
  border-radius: 50%;
  margin: 0 auto;
  display: block;
}
```

register.css

> frontend/src/pages/stylesheets/register.css

Written by Eugene Au

```
/* center the content */
#display {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  width: 100vw;
}

/* adding padding */
#register-wrapper {
  width: calc(100% - 40px);
}

/* the form */
#register {
  position: absolute;
  width: inherit;

  /* center the content from top to bottom */
  max-height: 100vh;
  transform: translateY(-50%);
  top: 50%;
}

/* the spacer for header and footer */
#spacer {
  padding: 70px 0;
  margin-bottom: 120px;
  overflow: scroll;

  max-width: 500px;
  min-width: 200px;
```

```
  margin-left: auto;
  margin-right: auto;
}

/* The title of the page */
#register #title {
  font-size: 36px;
  font-weight: 700;
  color: black;
  display: flex;
  justify-content: center;
  align-items: center;
}

/* the forms and button  */
#register #form {
  /* center the content from top to bottom */
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;

  /* margin from the title */
  margin-top: 40px;
}

/* each input field */
#register #form .text.input {
  /* make the input field take up the whole width */
  width: 100%;

  /* styles */
  font-family: Outfit, sans-serif;
  font-size: 16px;
  border: none;
  border-radius: 100px;
  padding: 13px 21px;
  font-weight: 600;
  color: black;
  background-color: #e7e7e7;

  /* make the form start from the left */
  box-sizing: border-box;

  /* Remove the red border when selected */
  outline: none;
}

/* text input field's spacing */
```

```css
#register #form .field {
  margin-bottom: 30px;
}

/* error message */
label.error {
  color: red;
  font-size: 14px;
  margin-top: 5px;
  margin-bottom: 10px;
  margin-left: 5px;
  display: flex;
}

#form .checkbox {
  /* make the checkbox center */
  color: black;
  display: flex;
  align-items: center;
  line-height: 1.3;
}

#form input[type="checkbox"] {
  width: 21px;
  height: 21px;
  margin: 11px;
  border-radius: 12px;
}

#form button {
  width: 100%;

  /* styles */
  font-size: 16px;
  font-weight: 700;
  border: none;
  border-radius: 100px;
  padding: 10px;
  color: black;
  background-color: #00dca5;
  margin-bottom: 70px;
  font-family: "Outfit";
}

#form .text {
  color: black;
}
```

serverError.css

> frontend/src/pages/stylesheets/serverError.css

Written by Eugene Au

```
#error {
  padding: 70px;
  color: red;
}
```

test.css

> frontend/src/pages/stylesheets/test.css

Written by Eugene Au

```
.test button {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
```

Stores

geoLocationStore.ts

> frontend/src/pages/stores/geoLocationStore.ts

Written by Adam George

```
import { create } from "zustand";

type Position = {
    lat: number;
    lng: number;
}

type positionStore = {
    position: Position;
    setPosition: (lat: number, lng: number) => void;
}

type geoTagStore = {
```

```
    geoTag: string;
    setGeoTag: (tag: string) => void;
}

export const usePositionStore = create<positionStore>((set) => ({
    position: {lat: 0, lng: 0},
    setPosition: (lat, lng) => set({position: {lat, lng}})
}));

export const useGeoTagStore = create<geoTagStore>((set) => ({
    geoTag: "Unknown Location",
    setGeoTag: (tag) => set({geoTag: tag})
}));
```

pinStore.ts

> frontend/src/pages/stores/pinStore.ts

Written by Adam George

```
import { create } from "zustand";

// Placeholder imports
const image1 =

"https://cdn.discordapp.com/attachments/1204728741230809098/1207497297022160978/1000
016508.JPG?
ex=65dfdc7d&is=65cd677d&hm=295b9625886c4e12ea212d291878bb71d37e22a31d71e5757546d0a4a
0a1bdb4&";
const image2 =

"https://cdn.discordapp.com/attachments/1204728741230809098/1207497297961943050/1000
015958.JPG?
ex=65dfdc7e&is=65cd677e&hm=6413142376bf1efe664ef897cd70325b1f5f2d03e9d177ab4b9c541a5
fb1de59&";
const image3 =

"https://cdn.discordapp.com/attachments/1204728741230809098/1207497298116874311/1000
016354.JPG?
ex=65dfdc7e&is=65cd677e&hm=e497673ab0de533871fc5fb4bb6e702ce4fbaa856f99461dc3abf555c
6f0d510&";

type Pin = {
    id: number;
    position: { lat: number; lng: number };
    caption: string;
    image: string;
    datetime: string;
```

```
    open: boolean;
    collected: boolean;
}

type pinStore = {
    pins: Pin[];
    setPins: (pins: Pin[]) => void;
    addPin: (pin: Pin) => void;
    removePin: (id: number) => void;
}

type pinId = {
    pinIds: number[];
}

export const usePinStore = create<pinStore>((set) => ({
    pins: [],
    setPins: (pins) => set({pins: pins.map(pin => ({...pin, collected: false}))}),
    addPin: (pin) => set((state) => ({pins: [...state.pins, pin]})),
    removePin: (id) => set((state) => ({pins: state.pins.filter((pin) => pin.id !==
id)}))
}));

export const useCollectedPinStore = create<pinId>((set) => ({
    pinIds: [],
    setPinIds: (pinIds: number[]) => set({pinIds: pinIds}),
    addPinId: (id: number) => set((state) => ({pinIds: [...state.pinIds, id]})),
}));
```