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

Backend files

Database

test_models.py

> backend/PostCard/database/test_models.py Written by Benjamin Ellision

```
from django.test import TestCase
from database import models


class GeolocationTestCase(TestCase):

    def setUp(self):
        models.Geolocation.objects.create(location="TEST
LOCATION",latitude=0.0,longitude=0.0)

    def test_geolocation(self):
        """Testing Geolocation Generation Data"""
        testLoc = models.Geolocation.objects.get(location="TEST LOCATION")
        self.assertEqual(testLoc.latitude, 0.0)
        self.assertEqual(testLoc.longitude, 0.0)

    def tearDown(self):
        testLoc = models.Geolocation.objects.get(location="TEST LOCATION")
        testLoc.delete()



class PostsTestCase(TestCase):
    def setUp(self):
        exampleLoc = models.Geolocation.objects.create(location="TEST
LOCATION",latitude=0.0,longitude=0.0)
        exampleUser = models.User.objects.create(username="TEST",email="T@E.ST",
password="TEST")
        models.Posts.objects.create(image="image.jpg",
geolocID=exampleLoc,userid=exampleUser,caption="TEST")

    def test_post(self):
        """Testing Post Data"""
        from datetime import datetime
        import pytz
        timezone = pytz.timezone('Europe/London')
        currentDate = datetime.now(timezone).date()
        testPost =
models.Posts.objects.get(userid=models.User.objects.get(username="TEST"))
        self.assertEqual(testPost.datetime.date(), currentDate)
```

```
    def tearDown(self):

models.Posts.objects.get(userid=models.User.objects.get(username="TEST")).delete()
        models.Geolocation.objects.get(location="TEST LOCATION").delete()
        models.User.objects.get(username="TEST").delete()


class StickersTestCase(TestCase):

    def setUp(self):
        models.Stickers.objects.create(stickersName="TEST STICKER", fileName="TEST
FILE NAME")

    def test_Sticker(self):
        testSticker = models.Stickers.objects.get(stickersName="TEST STICKER")
        self.assertEqual(testSticker.stickerPrice, 25)
        self.assertEqual(testSticker.stickersDescription, "sticker")

    def tearDown(self):
        testSticker = models.Stickers.objects.get(stickersName="TEST STICKER")
        testSticker.delete()



class PostsUserTestCase(TestCase):

    def setUp(self):
        exampleSticker = models.Stickers.objects.create(stickersName="TEST
STICKER", fileName="TEST FILE NAME")
        exampleLoc = models.Geolocation.objects.create(location="TEST
LOCATION",latitude=0.0,longitude=0.0)
        exampleUser = models.User.objects.create(username="TEST",email="T@E.ST",
password="TEST")
        examplePost = models.Posts.objects.create(image="image.jpg",
geolocID=exampleLoc,userid=exampleUser,caption="TEST")
        examplePostUser = models.PostsUser.objects.create(
            userID = exampleUser
        )

    def test_userdata(self):
        """Testing User Data"""
        exampleUser = models.User.objects.get(username="TEST")
        exampleUserData = models.PostsUser.objects.get(userID=exampleUser)
        self.assertEqual(exampleUserData.coins, 0)
        self.assertEqual(str(exampleUserData.postID), "database.Posts.None")
        self.assertEqual(str(exampleUserData.unlockedAvatars),
"database.Stickers.None")
        self.assertEqual(str(exampleUserData.avatarInUse), "None")
        self.assertEqual(exampleUserData.postsMade, 0)
        self.assertEqual(exampleUserData.postsSaved, 0)
        self.assertEqual(exampleUserData.postsMadeToday, 0)
        self.assertEqual(exampleUserData.postsSavedToday, 0)
        self.assertEqual(exampleUserData.youtubeLink, "")
        self.assertEqual(exampleUserData.twitterLink, "")
```

```
        self.assertEqual(exampleUserData.instagramLink, "")
        self.assertEqual(exampleUserData.bio, "")

    def tearDown(self):
        exampleUser = models.User.objects.get(username="TEST")
        models.PostsUser.objects.get(userID=exampleUser).delete()
        models.Posts.objects.get(userid=exampleUser).delete()
        exampleUser.delete()
        models.Geolocation.objects.get(location="TEST LOCATION").delete()
        models.Stickers.objects.get(stickersName="TEST STICKER").delete()




class ChallengesTestCase(TestCase):

    def setUp(self):
        models.Challenges.objects.create(challengeDesc="TEST CHALLENGE")

    def test_challenge(self):
        testChallenge = models.Challenges.objects.get(challengeDesc="TEST
CHALLENGE")
        self.assertEqual(testChallenge.postsNeeded, 9999999)
        self.assertEqual(testChallenge.savesNeeded, 9999999)
        self.assertEqual(testChallenge.inUse, False)
        self.assertEqual(testChallenge.type, "noneType")
        self.assertEqual(testChallenge.coinsRewarded, 0)

    def tearDown(self):
        testChallenge = models.Challenges.objects.get(challengeDesc="TEST
CHALLENGE")
        testChallenge.delete()
```

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'
```

admin.py

```
from django.contrib import admin
from .models import Geolocation, Posts, Stickers, StickersUser, PostsUser,
Challenges, CurrentDay
```

```
# 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')

class ChallengesAdmin(admin.ModelAdmin):
    list_display = ('id', 'challengeDesc', 'inUse')

class CurrentDayAdmin(admin.ModelAdmin):
    list_display = ('id', 'dateOfLastInteraction')

# 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)
admin.site.register(Challenges, ChallengesAdmin)
admin.site.register(CurrentDay, CurrentDayAdmin)
```

models.py

> backend/PostCard/database/models.py Written by Ziyad Alnawfal,Eugene Au,Ben Ellision,Jayant Chawla

```
from django.db import models
from django.core.validators import MaxValueValidator
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

# TODO
#   Rename table to "Avatars"
#   Rename stickersName to avatarName
#   Delete stickersDescription

class Stickers(models.Model):
    id   = models.AutoField(primary_key=True)
    stickersName = models.CharField(max_length = 50)
    stickerPrice = models.IntegerField(default = 25)
    stickersDescription = models.CharField(default ="sticker",max_length = 100)
    fileName = models.CharField(max_length = 100)

# TODO
#   Use for sticker-user association for a lighter UserData table
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)

# TODO
#   Rename table to UserData
class PostsUser(models.Model):
    userID = models.OneToOneField(User,  on_delete = models.CASCADE)
    coins = models.PositiveIntegerField(default=0)
    postID = models.ManyToManyField(Posts,blank=True)
    unlockedAvatars = models.ManyToManyField(Stickers, blank=True
,related_name="unlocked")
    avatarInUse = models.ForeignKey(Stickers,related_name="profile_pic", on_delete
= models.CASCADE, null=True, blank=True)
    postsMade = models.PositiveIntegerField(default=0)
    postsSaved = models.PositiveIntegerField(default=0)
    postsMadeToday = models.PositiveSmallIntegerField(default=0, validators=
[MaxValueValidator(48)])
    postsSavedToday = models.PositiveSmallIntegerField(default=0)
    youtubeLink = models.CharField(max_length = 255, default="",blank = True)
    twitterLink = models.CharField(max_length = 255, default="",blank = True)
    instagramLink = models.CharField(max_length = 255,default="", blank = True)
    bio = models.CharField(max_length = 255, default="",blank = True)


class Challenges(models.Model):
```

```
    id = models.AutoField(primary_key=True)
    challengeDesc = models.CharField(max_length = 100)
    postsNeeded = models.PositiveSmallIntegerField(default=9999999)
    savesNeeded = models.PositiveSmallIntegerField(default=9999999)
    inUse = models.BooleanField(default=False)
    type = models.CharField(max_length=20, blank = True, default="noneType")
    coinsRewarded = models.PositiveIntegerField(default=0)

class CurrentDay(models.Model):
    dateOfLastInteraction = models.DateField(default="1111-11-11")
```

PostCard

asgi.py

> backend/PostCard/PostCard/asgi.py

```
"""
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,
Challenges, CurrentDay
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

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

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

settings.py

> backend/PostCard/PostCard/settings.py Written by Ziyad Alnawfal,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'
```

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()
```

urls.py

> backend/PostCard/PostCard/wsgi.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)
```

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'
```

checkWinner.py

> backend/PostCard/posts/checkWinner.py Written by Ziyad Alnawfal

```
def checkWinner(user_info,challenge):
    if challenge.type == "daily":
        if challenge.postsNeeded == 0:
            if challenge.savesNeeded == user_info.postsSavedToday:
                user_info.coins += challenge.coinsRewarded
        else:
            if challenge.postsNeeded == user_info.postsMadeToday:
                user_info.coins += challenge.coinsRewarded

    elif challenge.type == "milestone":
        if challenge.savesNeeded == 0:
            if challenge.postsNeeded == user_info.postsMade:
                user_info.coins += challenge.coinsRewarded
        else:
            if challenge.savesNeeded == user_info.postsSaved:
                user_info.coins += challenge.coinsRewarded

    user_info.save()
```

daily_reset.py

> backend/PostCard/posts/daily_reset.py Written by Benjamin Ellision

```
# function that resets user info related to all daily challenges and actives a
random daily challenge
def dailyReset():
    try:
        import random
        from datetime import datetime
        import pytz
        from database.models import PostsUser, Challenges, CurrentDay
        timezone = pytz.timezone('Europe/London')
        currentDate = datetime.now(timezone).date()
        print("CURRENT DATE ACCORDING TO DEVICE = " + str(currentDate))
        date,_ = CurrentDay.objects.get_or_create()
```

```
        print("CURRENT DATE ACCORDING TO DATABASE = " +
str(date.dateOfLastInteraction))

        # checking if the daily challenge has expired, (24hrs)
        if currentDate != date.dateOfLastInteraction:
            random.seed()
            # reseting user info for daily challenges
            for x in PostsUser.objects.all():
                x.postsMadeToday=0
                x.postsSavedToday=0
                x.save()

            # Resets all daily challenges to inactive and actives a random one
            number_daily_challenges=0
            for y in Challenges.objects.filter(type="daily"):
                y.inUse=False
                number_daily_challenges+=1
                y.save()
            z=Challenges.objects.filter(type="daily")
[random.randint(0,number_daily_challenges)]
            z.inUse=True
            z.save()

            date.dateOfLastInteraction = currentDate
            date.save()
            print("DAILY RESET SUCCESSFUL")
        print("DAILY RESET NOT NEEDED")
    except Exception as err:
        print(err)
```

test_views.py

> backend/PostCard/posts/test_views.py Written by Jayant Chawla

```
from django.test import TestCase
from django.urls import reverse
from rest_framework import status
from rest_framework.test import APIClient
from database.models import Geolocation, Posts, Stickers, PostsUser, Challenges
from django.contrib.auth.models import User
from .views import *
from django.core.files.uploadedfile import SimpleUploadedFile

# USE follwing code to run: python  manage.py test --pattern="test_views.py"



class PostCardTests(TestCase):
    def setUp(self):
        # Create a test client for making API requests
        self.client = APIClient()
```

```
        self.user1 = User.objects.create_user(username='user1', password='pass')
        self.user2 = User.objects.create_user(username='user2', password='pass')
        self.geolocation = Geolocation.objects.create(location='Test Location',
latitude=10.0, longitude=20.0)
        self.super_user = User.objects.create_superuser('superuser',
'superuser@example.com', 'superpassword')
        self.user_to_delete = User.objects.create_user(username='todelete',
password='testpass')
        self.client.force_authenticate(user=self.user1)
        self.geolocation = Geolocation.objects.create(location='Test Location',
latitude=10.0, longitude=20.0)
        self.post = Posts.objects.create(caption='Test Post',
geolocID=self.geolocation, userid=self.user1)
        self.sticker = Stickers.objects.create(stickersName='Test Sticker',
stickerPrice=0, fileName='NULL')
        self.sticker1 = Stickers.objects.create(stickersName='Avatar1',
stickerPrice=100, fileName='NULL')
        self.sticker2 = Stickers.objects.create(stickersName='Avatar2',
stickerPrice=100, fileName='NULL')
        self.postUser = PostsUser.objects.create(userID=self.user1, bio='Test
Bio')
        self.challenge = Challenges.objects.create(postsNeeded=5,
coinsRewarded=25, challengeDesc="Create 5 posts", type="daily")
        self.postUser.unlockedAvatars.add(self.sticker1)
        self.daily_challenge = Challenges.objects.create(
            postsNeeded=5, savesNeeded=0, coinsRewarded=25, challengeDesc="Create
5 posts", type="daily", inUse=True)
        self.milestone_challenge_1 = Challenges.objects.create(
            postsNeeded=35, savesNeeded=0, coinsRewarded=250,
challengeDesc="Create 35 posts", type="milestone", inUse=True)
        self.milestone_challenge_2 = Challenges.objects.create(
            postsNeeded=0, savesNeeded=35, coinsRewarded=250, challengeDesc="Save
35 posts", type="milestone", inUse=True)
        self.posts_user = PostsUser.objects.get(userID=self.user1)
        self.posts_user.postID.add(self.post)


    def test_get_avatar(self):
        # Test getting all posts
        response = PostsUser.objects.get(userID=self.user1)
        print(response)


    def test_get_posts(self):
        # Test getting all posts
        response = self.client.get(reverse('posts-list'))
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertEqual(len(response.data), 1)
        print(response, response.data)

    def test_create_post(self):
    # Use a real image file's path or include a valid image file in your test
directory
        image_path = r'./media/media/Avatar/crown.png'
```

```
        with open(image_path, 'rb') as img:
            image_data = img.read()

        image = SimpleUploadedFile('test_image.png', image_data,
content_type='image/png')
        data = {
            'userid': self.user1.id,
            'caption': 'A new post',
            'geolocID': self.geolocation.id,
            'image': image,
        }

        self.client.force_authenticate(user=self.user1)
        response = self.client.post(reverse('posts-list'), data,
format='multipart')
        print(response.data)
        self.assertEqual(response.status_code, status.HTTP_201_CREATED)


    def test_change_bio(self):
        # Login as user1 to perform the bio update
        self.client.force_authenticate(user=self.user1)
        # Define the new bio and social media links
        new_bio_data = {
            'bio': 'Updated Bio',
            'youtube': 'http://youtube.com/newuser1',
            'twitter': 'http://twitter.com/newuser1',
            'instagram': 'http://instagram.com/newuser1',
        }
        # Make a POST request to the changeBio endpoint
        response = self.client.post(reverse('change-bio'), new_bio_data,
format='json')
        # Assert that the response status code is HTTP 200 OK
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        # Fetch the updated PostsUser object from the database
        updated_user_info = PostsUser.objects.get(userID=self.user1.id)
        # Assert that the PostsUser object has been updated with the new bio and
social media links
        self.assertEqual(updated_user_info.bio, new_bio_data['bio'])
        self.assertEqual(updated_user_info.youtubeLink, new_bio_data['youtube'])
        self.assertEqual(updated_user_info.twitterLink, new_bio_data['twitter'])
        self.assertEqual(updated_user_info.instagramLink,
new_bio_data['instagram'])
        print(response, response.data)

    # def test_get_user_profile(self):
    #     # Authenticate as user1
    #     self.client.force_authenticate(user=self.user1)

    #     # Make a GET request to the 'get-user' endpoint
    #     response = self.client.get(reverse('user-list'))

    #     # Verify the response status code is 200 OK
    #     self.assertEqual(response.status_code, status.HTTP_200_OK)
```

```
    #     # Check if the response contains the expected data
    #     expected_fields = ['username', 'coins', 'profilePicture', 'Bio',
'youtube', 'instagram', 'twitter']
    #     for field in expected_fields:
    #         self.assertIn(field, response.data)

    #     # Optionally, verify the values of certain fields
    #     self.assertEqual(response.data['username'], self.user1.username)
    #     # You can add more assertions here to verify other fields like 'coins',
'Bio', etc.


    def test_get_all_users(self):
        # Authenticate as the superuser
        self.client.force_authenticate(user=self.super_user)
        # Make the GET request to the 'get-all-users' endpoint
        response = self.client.get(reverse('all-users'))
        # Check that the response status code is 200 OK
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        expected_user_count = User.objects.count()
        self.assertEqual(len(response.data), expected_user_count)
        super_user_data = next((item for item in response.data if item["username"]
== "superuser"), None)
        self.assertIsNotNone(super_user_data)
        self.assertTrue(super_user_data['is_superuser'])
        print(response, response.data)

    def test_get_all_avatars(self):
        # Make a GET request to the 'get-all-avatars' endpoint
        response = self.client.get(reverse('get-all-avatars'))
        # Check that the response status code is 200 OK
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        # Check the response contains the avatar owned by user1
        owned_avatar_names = [avatar['name'] for avatar in response.data]
        self.assertIn('Avatar1', owned_avatar_names)
        self.assertNotIn('Avatar2', owned_avatar_names)  # Assuming user1 does not
own Avatar2
        print(response)


    def test_get_avatars(self):
        # Make a GET request to the 'get-avatars' endpoint
        response = self.client.get(reverse('get-avatars'))
        # Check that the response status code is 200 OK
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        # Check the response contains avatars NOT owned by user1 (e.g., Avatar2)
        not_owned_avatar_names = [avatar['name'] for avatar in response.data]
        self.assertNotIn('Avatar1', not_owned_avatar_names)  # User1 owns Avatar1,
so it should not be in the list
        self.assertIn('Avatar2', not_owned_avatar_names)  # Assuming user1 does
not own Avatar2
        print(response)
```

```
    def test_get_challenges(self):
        # Authenticate as user1
        self.client.force_authenticate(user=self.user1)
        # Make a GET request to the 'get-challenges' endpoint
        response = self.client.get(reverse('get-challenges'))
        # Verify the response status code is 200 OK
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        # Check if the response contains the expected data
        self.assertIn('DailyChallenge', response.data)
        self.assertIn('DailyCoinsRewarded', response.data)
        self.assertIn('Milestone1Challenge', response.data)
        self.assertIn('Milestone1CoinsRewarded', response.data)
        self.assertIn('Milestone2Challenge', response.data)
        self.assertIn('Milestone2CoinsRewarded', response.data)
        self.assertEqual(response.data['DailyChallenge'],
self.daily_challenge.challengeDesc)
        self.assertEqual(response.data['Milestone1Challenge'],
self.milestone_challenge_1.challengeDesc)
        self.assertEqual(response.data['Milestone2Challenge'],
self.milestone_challenge_2.challengeDesc)
        print(response, response.data)


    def tearDown(self):
        super().tearDown()
        PostsUser.objects.filter(userID=self.user1).delete()
        for post in Posts.objects.all():
            if post.image:
                post.image.delete(save=False)  # This deletes the file from storag



    # def test_add_collection(self):
    #     # Simulate adding a post to the user's collection
    #     response = self.client.post(reverse('add-collection'), {'postid':
self.post.id}, format='json')

    #     # Check that the response indicates success
    #     self.assertEqual(response.status_code, status.HTTP_201_CREATED)
    #     self.assertIn("Post added to collection", response.data["message"])

    #     # Verify the post is added to the user's collection
    #     user_collection = PostsUser.objects.get(userID=self.user).postID.all()
    #     self.assertIn(self.post, user_collection)

    #     # Optionally, check challenge updates if relevant
    #     user_profile = PostsUser.objects.get(userID=self.user)
    #     self.assertEqual(user_profile.postsSavedToday, 1)
    #     self.assertEqual(user_profile.postsSaved, 1)


    def test_get_collections_with_saved_posts(self):
        response = self.client.get(reverse('collected-posts'))
        self.assertEqual(response.status_code, status.HTTP_200_OK)
```

```
        self.assertEqual(len(response.data), 1)  # Expecting one saved post
        self.assertEqual(response.data[0]['caption'], 'Test Post')  # Ensure the
correct post is returned
        print(response, response.data)

    def test_get_collections_without_saved_posts(self):
        # Remove all saved posts for this test
        self.posts_user.postID.clear()
        response = self.client.get(reverse('collected-posts'))
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertEqual(len(response.data), 0)  # Expecting no saved posts
        print(response, response.data)

    def test_delete_post(self):
        # Authenticate as a superuser since the view requires IsSuperUser
permission
        self.client.force_authenticate(user=self.super_user)
        # Create a post to delete
        post_to_delete = Posts.objects.create(caption='Delete Me',
geolocID=self.geolocation, userid=self.user1)
        # Make the DELETE request to the 'delete-post' endpoint with the post's id
        response = self.client.delete(reverse('deletePost', kwargs={'pk':
post_to_delete.id}))
        # Check that the response indicates success (HTTP 204 No Content)
        self.assertEqual(response.status_code, status.HTTP_204_NO_CONTENT)
        with self.assertRaises(Posts.DoesNotExist):
            Posts.objects.get(pk=post_to_delete.id)
        print(response, response.data)

    def test_delete_user(self):
        # Authenticate as the superuser
        self.client.force_authenticate(user=self.super_user)
        # Make the DELETE request to the 'delete-user' endpoint with the user's id
to be deleted
        response = self.client.delete(reverse('deleteUser', kwargs={'pk':
self.user_to_delete.pk}))
        # Check that the response indicates success (HTTP 204 No Content)
        self.assertEqual(response.status_code, status.HTTP_204_NO_CONTENT)
        with self.assertRaises(User.DoesNotExist):
            User.objects.get(pk=self.user_to_delete.pk)
        print(response, response.data)
```

urls

> backend/PostCard/posts/urls.py Written by Ziyad Alnawfal, 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'),
    path('logout/',UserLogout, name="logout"),
    #-----
    #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'),

    #CHALLENGES API ENDPOINT
    path('getChallenges/',getChallenges, name='get-challenges'),
    path('getAvatars/',getAvatars,name="get-avatars"),
    path('changeAvatar/',changeAvatar,name="change-avatar"),
    path('getAllAvatars/',getAllAvatars,name="get-all-avatars"),
    path('createObjects/',createObjects,name="create-Objects"),
    path('purchase/',purchase, name ="purchase"),
    path('changeBio/',changeBio,name="change-bio"),
    path('createObjects/',createObjects,name="create-objects"),

    path('deletePost/<int:pk>/', deletePost, name='deletePost'),
    path('deleteUser/<int:pk>/', deleteUser, name='deleteUser'),

    #USER API ENDPOINT
    path('getUser/', getUser, name='user-list'),
    path('isSuperUser/', checkSuperuser, name='is-superuser'),

    #ADMIN API ENDPOINT
    path('getAllPosts/', getPosts, name='get-posts'),
    path('getAllUsers/', getAllUsers, name='all-users'),
    path('deletePost/<int:pk>/', deletePost, name='deletePost'),
    path('deleteUser/<int:pk>/', deleteUser, name='deleteUser'),
    path('checkSuperUserId/<int:user_id>/', checkSuperuserById, name='check-
```

```
superuser-id'),
] + 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, Jayant Chawla, Eugene Au

```
from django.shortcuts import render
from rest_framework import generics
from database.models import Geolocation, Posts, Stickers, PostsUser, Challenges
from PostCard.serializers import PostsSerializer, GeolocationSerializer,
StickersSerializer,StickersUser, StickersUserSerializer, PostsUserSerializer,
ChallengesSerializer
from rest_framework.permissions import AllowAny, IsAuthenticated, IsAdminUser
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
from .daily_reset import dailyReset
from django.conf import settings
from django.urls import path
import os
from django.db import IntegrityError
from .checkWinner import checkWinner
from django.contrib.auth.models import User
from django.contrib.auth.decorators import login_required

# Custom permission class to check if the user is a superuser
class IsSuperUser(IsAdminUser):
    def has_permission(self, request, view):
        return bool(request.user and request.user.is_superuser)

# 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 PostUserList(generics.ListCreateAPIView):
    queryset = PostsUser.objects.all()
    serializer_class = PostsUserSerializer
    permission_classes = [AllowAny]

#Returns a specific post made by a user
class PostUserDetail(generics.RetrieveAPIView):
    queryset = PostsUser.objects.all()
    serializer_class = PostsUserSerializer
    permission_classes = [AllowAny]

#Returns all Challenges
class ChallengesList(generics.ListCreateAPIView):
    queryset = Challenges.objects.all()
    serializer_class = ChallengesSerializer
    permission_classes = [AllowAny]

#Returns a specific challenge
class ChallengesDetail(generics.RetrieveAPIView):
```

```
    queryset = Challenges.objects.all()
    serializer_class = ChallengesSerializer
    permission_classes = [AllowAny]

@api_view(['POST','Get'])
@permission_classes([AllowAny])
#CREATES ALL OBJECTS NEEDED , MUST BE CALLED FIRST
def createObjects(request):
    try:
        #create a null sticker
        Stickers.objects.get_or_create(stickersName="default",stickerPrice
=0,fileName="NULL")
        # Creating all challenges
        #Daily
        x,_ = Challenges.objects.get_or_create(postsNeeded = 5, coinsRewarded =
25, challengeDesc="Create 5 posts",type="daily")
        y,_ = Challenges.objects.get_or_create(savesNeeded = 5, coinsRewarded =
25, challengeDesc ="Save 5 posts",type="daily")
        #Milestone
        a,_ = Challenges.objects.get_or_create(postsNeeded = 35, inUse = True,
coinsRewarded = 250, challengeDesc = "Create 35 posts",type="milestone")
        b,_ = Challenges.objects.get_or_create(savesNeeded = 35, inUse = True,
coinsRewarded = 250, challengeDesc = "Save 35 posts",type="milestone")

        base = f"{request.scheme}://{request.get_host()}
{settings.MEDIA_URL}media/avatar/"
        avatar_files = os.listdir(os.path.join(settings.MEDIA_ROOT,
'media/avatar'))

        #Loops through all the avatar files and appending it to base
        for files in avatar_files:
            index = files.index(".")
            # Creating all the needed sticker objects
            x,y = Stickers.objects.get_or_create(stickersName = files[:index],
fileName = base+files, stickerPrice = 25)
            if y == False: # sticker does not exist
                x.save()

    except Exception as e:
        return Response({e},status=status.HTTP_404_NOT_FOUND)

    return Response({"Database set up"},status=status.HTTP_201_CREATED)

@api_view(['POST','Get'])
@permission_classes([AllowAny])
# view to change a user's Bio
def changeBio(request):
    try:
        user = request.user.id
        bio = request.data['bio']
        youtubeUrl = request.data['youtube']
        twitterUrl = request.data['twitter']
        instagramUrl = request.data['instagram']
```

```
            user_info,_ = PostsUser.objects.get_or_create(userID=user)
            user_info.bio = bio
            user_info.youtubeLink = youtubeUrl
            user_info.twitterLink = twitterUrl
            user_info.instagramLink = instagramUrl

            user_info.save()

        except Exception as e:
            return Response({e},status=status.HTTP_400_BAD_REQUEST)
        return
Response({"youtube":youtubeUrl,"twitter":twitterUrl,"instagram":instagramUrl,"Bio"
:bio},status=status.HTTP_200_OK)

@api_view(['POST']) # Secuirty purposes we do not want to append user details to
header
@permission_classes([AllowAny])
# view for creating a post
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:]
    request.data['userid'] = userid  # Add 'userid' key

    try:
        userData = PostsUser.objects.get(userID = userid)
        milestone_1 = Challenges.objects.get(postsNeeded=35)
        Inuse_challenges = Challenges.objects.filter(inUse=True)

        # Finding the daily challenge
        for i in Inuse_challenges:
            if i.postsNeeded < 10 and i.savesNeeded < 10:
                todays_challenge = i

        # Conditionals for make sure checkWinner is not called when a user has
already met the challenge's requirement
        if userData.postsMadeToday != todays_challenge.postsNeeded:
            userData.postsMadeToday += 1
            userData.save()
            checkWinner(userData,todays_challenge)

        if userData.postsMade != milestone_1.postsNeeded:
```

```
            userData.postsMade += 1
            userData.save()
            checkWinner(userData,milestone_1)

        dailyReset()
    except:
        print("Failed to get user data")

    # Create a serializer instance with the mutable copy of request.data
    serialized = PostsSerializer(data=request.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(['GET'])
@permission_classes([IsAuthenticated])
def checkSuperuser(request):
    is_superuser = request.user.is_superuser
    return Response({"is_superuser": is_superuser}, status=status.HTTP_200_OK)

@api_view(['GET'])
@permission_classes([IsSuperUser])
def checkSuperuserById(request, user_id):
    try:
        user = User.objects.get(id=user_id)
        is_superuser = user.is_superuser
        return Response({"is_superuser": is_superuser}, status=status.HTTP_200_OK)
    except User.DoesNotExist:
        return Response({"error": "User does not exist"},
status=status.HTTP_404_NOT_FOUND)

@api_view(['POST' ,'Get'])
@permission_classes([AllowAny])
# view for returning all user related Info
def getUser(request):
    try:
        dailyReset()
        user = User.objects.get(id=request.user.id)
        user_info, _ = PostsUser.objects.get_or_create(userID=user)
        username = user.username

        if user_info.unlockedAvatars.exists() == False:

user_info.unlockedAvatars.add(Stickers.objects.get(stickersName="default"))
            user_info.avatarInUse =
user_info.unlockedAvatars.get(stickersName="default")
            user_info.save()

    except Exception as e:
        print(e)
```

```
        # if user is not logged in, then raise an error
        return Response({"Message"},status=status.HTTP_400_BAD_REQUEST)

    return Response({"username":username,"coins":user_info.coins,"profilePicture":
user_info.avatarInUse.fileName,

"Bio":user_info.bio,"youtube":user_info.youtubeLink,"instagram":user_info.instagra
mLink,"twitter":user_info.twitterLink},status=status.HTTP_200_OK)

@api_view(['GET'])
@permission_classes([IsSuperUser])
def getAllUsers(request):
    try:
        data = []

        # Prepare the data to return
        for user in User.objects.all():
            user_info, _ = PostsUser.objects.get_or_create(userID=user)

            profilePicture = user_info.avatarInUse.fileName if
user_info.avatarInUse and user_info.avatarInUse.fileName else 'default.jpg'

            data.append({
            'id': user.id,
            'username': user.username,
            'profilePicture': profilePicture,
            'bio': user_info.bio,
            'youtube':user_info.youtubeLink,
            'instagram':user_info.instagramLink,
            'twitter':user_info.twitterLink,
            'is_superuser': user.is_superuser,
            }
            )
            # 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','GET'])
@permission_classes([AllowAny])
# view for changing a user's avatar in their profile
def changeAvatar(request):
    user = request.user.id
    user_info,_ = PostsUser.objects.get_or_create(userID = user)
    profile_name = request.data['avatar'] # this correct?

    unlocked_avatars = user_info.unlockedAvatars.all()
    unlocked_avatars_list = []
    for avatars in unlocked_avatars:
        unlocked_avatars_list.append(avatars.stickersName)

    if profile_name not in unlocked_avatars_list:
        return Response({"You have not unlocked this
```

```
avatar"},status=status.HTTP_401_UNAUTHORIZED)
    else:
        profile_pic = Stickers.objects.get(stickersName = profile_name)
        user_info.avatarInUse = profile_pic
        user_info.save()


        return Response({"you successfully changed your profile! Your current
profile is": user_info.avatarInUse.fileName, "Avatar name is":
user_info.avatarInUse.stickersName},status=status.HTTP_200_OK)

@api_view(['POST','GET'])
@permission_classes([AllowAny])
# RETURNS ALL AVATARS NOT OWNED BY A USER
def getAvatars(request):
    try:
        user = request.user
        user_info,_ = PostsUser.objects.get_or_create(userID=user)

        # loops and returns all avatars a user owns in a list
        all_avatars = user_info.unlockedAvatars.all()
        all_avatars_list = []
        for avatars in all_avatars:
                all_avatars_list.append(avatars.stickersName)

        #loops and returns all avatars that a user DOES NOT own
        all_stickers = Stickers.objects.all()
        all_stickers_list = []
        for stickers in all_stickers:
            if stickers.stickersName not in all_avatars_list:

all_stickers_list.append({"name":stickers.stickersName,"price":stickers.stickerPri
ce,"path":stickers.fileName})

    except Exception as e:
        return Response({e}, status=status.HTTP_400_BAD_REQUEST)
    return Response(all_stickers_list,status=status.HTTP_200_OK)


@api_view(['POST','GET'])
@permission_classes([AllowAny])
# RETURNS ALL AVATARS OWNED BY A USER
def getAllAvatars(request):
    user = request.user.id
    user_info = PostsUser.objects.get(userID=user)

    all_avatars = user_info.unlockedAvatars.all()
    all_avatars_list = []

    for avatars in all_avatars:

all_avatars_list.append({"name":avatars.stickersName,"price":avatars.stickerPrice,
"path":avatars.fileName})
    return Response(all_avatars_list,status=status.HTTP_200_OK)
```

```
@api_view(['POST' ,'Get'])
@permission_classes([AllowAny])
# view that returns all challenges active
def getChallenges(request):
    try:
        user = request.user.id
        user_info,_ = PostsUser.objects.get_or_create(userID=user)

        # gets list of objects that are active, loops and gets daily challenge
        Inuse_challenges = Challenges.objects.filter(inUse=True)
        for i in Inuse_challenges:
            if i.postsNeeded < 10 and i.savesNeeded < 10:
                todays_challenge = i

        milestone_1 = Challenges.objects.get(postsNeeded = 35)
        milestone_2 = Challenges.objects.get(savesNeeded = 35)


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

    # dict containing all challenge related info
    all_challenges = {"DailyChallenge":todays_challenge.challengeDesc,
"DailyCoinsRewarded":25,"Milestone1Challenge":milestone_1.challengeDesc
                      ,"Milestone1CoinsRewarded": milestone_1.coinsRewarded,
"Milestone2Challenge": milestone_2.challengeDesc,
"Milestone2CoinsRewarded":milestone_2.coinsRewarded}

    if todays_challenge.savesNeeded == 0:
        all_challenges["Daily"] = str(user_info.postsMadeToday) +"/"+
str(todays_challenge.postsNeeded)
    elif todays_challenge.postsNeeded == 0:
        all_challenges["Daily"] = str(user_info.postsSavedToday) +"/"+
str(todays_challenge.savesNeeded)

    if milestone_1.postsNeeded == 0:
        all_challenges["Milestone1"] =
str(user_info.postsSaved)+"/"+str(milestone_1.savesNeeded)
    elif milestone_2.savesNeeded == 0:
        all_challenges["Milestone2"] =
str(user_info.postsMade)+"/"+str(milestone_1.postsNeeded)
    else:
        all_challenges["Milestone1"] =
str(user_info.postsMade)+"/"+str(milestone_1.postsNeeded)
        all_challenges["Milestone2"] =
str(user_info.postsSaved)+"/"+str(milestone_2.savesNeeded)
    return Response(all_challenges, status=status.HTTP_200_OK)


@api_view(['POST' ,'Get'])
@permission_classes([AllowAny])
```

```
# view to allow users to buy avatars
def purchase(request):
    user = request.user
    user_data,_ = PostsUser.objects.get_or_create(userID=user)

    #Seeing if sticker exists
    try :
        sticker = request.data['sticker']
        avatar = Stickers.objects.get(stickersName=sticker) # getting the sticker
id assoicated with that avatar
    except:
        return Response({"Message": "Sticker does not
exist"},status=status.HTTP_409_CONFLICT)

    # Checks to see If user has enough coins to purchase a avatar
    if user_data.coins < avatar.stickerPrice:
        return Response({"Message": f"Insuffiecient funds, You require
{avatar.stickerPrice - user_data.coins} more coins"},
status=status.HTTP_409_CONFLICT)
    else:
        try:
            user_data.unlockedAvatars.add(avatar)
            user_data.coins -= avatar.stickerPrice
            user_data.save()
        except IntegrityError:
            return Response({"Message": "You already own this avatar"},
status=status.HTTP_409_CONFLICT)
        return Response({"Message": f"Successful purchase, you have
{user_data.coins}"}, status=status.HTTP_200_OK)

from django.utils import timezone
from datetime import timedelta

@api_view(['POST','GET'])
@permission_classes([AllowAny])
# view for displaying only posts made in the last 24hrs
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','GET'])
@permission_classes([IsSuperUser])
# gets all posts
def getPosts(request):
    try:
        data = []
        # Prepare the data to return
        for post in Posts.objects.select_related('geolocID'):

            data.append({
            'id': post.id,
            'userid': post.userid.id,
            'username': post.userid.username,
            'is_superuser': post.userid.is_superuser,
            '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])
# view for adding a post to the collection page
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)
    try:

        milestone_2 = Challenges.objects.get(savesNeeded=35)
        Inuse_challenges = Challenges.objects.filter(inUse=True)
        for i in Inuse_challenges:
            if i.postsNeeded < 10 and i.savesNeeded < 10:
                todays_challenge = i

        if posts_user.postsSavedToday != todays_challenge.savesNeeded:
            posts_user.postsSavedToday += 1
            posts_user.save()
            checkWinner(posts_user,todays_challenge)

        if posts_user.postsSaved != milestone_2.savesNeeded:
            posts_user.postsSaved += 1
            posts_user.save()
            checkWinner(posts_user,milestone_2)

        dailyReset()
    except:
        print("Failed to get user data")
    dailyReset()
    return Response({"message": "Post added to collection"},
status=status.HTTP_201_CREATED)



@api_view(['POST','GET'])
@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:
        return Response([], status=status.HTTP_200_OK)

    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)


@api_view(['DELETE'])
@permission_classes([IsSuperUser])
def deletePost(request, pk):
    try:
        post = Posts.objects.get(pk=pk)
        post.delete()
        return Response({"message": "Post deleted successfully"},
status=status.HTTP_204_NO_CONTENT)
    except Posts.DoesNotExist:
        return Response({"message": "Post not found"},
status=status.HTTP_404_NOT_FOUND)

@api_view(['DELETE'])
@permission_classes([IsSuperUser])
def deleteUser(request, pk):
    try:
        user = User.objects.get(pk=pk)
        # Optionally, delete related data if necessary
        # Posts.objects.filter(userid=user).delete()
        # PostsUser.objects.filter(userID=user).delete()
        # And any other related deletions
        user.delete()
        return Response({"message": "User and all related data deleted
successfully"}, status=status.HTTP_204_NO_CONTENT)
    except User.DoesNotExist:
        return Response({"message": "User not found"},
status=status.HTTP_404_NOT_FOUND)
```

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 Benjamin Ellision

```
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

```
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, Eugene Au

```
from rest_framework.decorators import api_view, permission_classes
from rest_framework.response import Response
from rest_framework import status
from .serializers import UserRegisterSerializer , UserLoginSerializer
from rest_framework.decorators import permission_classes
from rest_framework.permissions import *

from rest_framework.authtoken.models import Token
from django.contrib.auth import authenticate
from django.contrib.auth.models import User

@api_view(['POST']) # We only want to recieve POST requests here, GET REQUESTS ARE
INVALID!
@permission_classes([AllowAny])
def UserRegisterAuthentication(request):
    password_check = request.data.get("password")
    username_check = request.data.get("username")
    special_chars = ['!','@','#','$','%','^','_','-','?','/','*','+','-
','£','(','()',')','=','~','¬']  #List of special characters, add if you guys see
fit

    if password_check.isdigit():
        return Response({"Password error": "invalid password, Password cannot be
all ints"},status=status.HTTP_400_BAD_REQUEST)

    # No identical info in password
    elif username_check in password_check:
        return Response({"Password error": "invalid password, Cannot have your
```

```
username in your password"},status=status.HTTP_400_BAD_REQUEST)

    # Seeing if the password contains a special character
    special_char_found = False
    for chars in password_check:
        if chars in special_chars:
            special_char_found = True
            break

    if special_char_found == False:
        return Response({"Password error": "invalid password, Passwor must contain
a special char"},status=status.HTTP_400_BAD_REQUEST)

    # Seeing if password contains a capital letter
    char_is_upper = False
    for chars in password_check:
        if chars.isupper() == True:
            char_is_upper = True
            break

    if char_is_upper == False:
        return Response({"Password error": "invalid password, Passwor must contain
a capital letter"},status=status.HTTP_400_BAD_REQUEST)




    serializer = UserRegisterSerializer(data=request.data)
    if serializer.is_valid():

        # Retrieving validateed data
        username = serializer.validated_data.get('username')
        email = serializer.validated_data.get('email')
        password = serializer.validated_data.get('password')

        user_instance = User(username=username,email=email)
        user_instance.set_password(password) # Must use set password so django
recoginzes its a password
        user_instance.save()
        token, _ = Token.objects.get_or_create(user=user_instance) # Avoids
needing to login in after making page


        return Response({f"token": token.key},status=status.HTTP_200_OK)   #
Successful user creation
    else:
        # errors = str(serializer.errors) # Shows erros for: already used username
in db and an invalid email (no "@" and ending such as ".com")
        return Response(serializer.errors,status=status.HTTP_400_BAD_REQUEST) #
Failed user creation


@api_view(['POST']) # Secuirty purposes we do not want to append user details to
header
```

```
@permission_classes([AllowAny])
def UserLoginAuthentication(request):
    username = request.data.get('username')
    password = request.data.get('password')
    user = authenticate(username=username, password=password)

    if user: # checks to see if user exists in database

        token, _ = Token.objects.get_or_create(user=user)
        return Response({"token": token.key}, status=status.HTTP_200_OK)
    else:
        # if user does not exist
        return Response({"username": "Invalid credentials"},
status=status.HTTP_404_NOT_FOUND)

@api_view(['POST',"GET"])
@permission_classes([AllowAny])
def UserLogout(request):
    try:
        user = request.user
        user_token = Token.objects.get(user=user)
        user_token.delete() # Token for given user deleted
    except:
        return Response({"Message": "User does not exist"},
status=status.HTTP_400_BAD_REQUEST)
    return Response({"Message": "User logged out
successful"},status=status.HTTP_200_OK)
```

Frontend Files

Src

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>
);
```

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;
}
```

App.jsx

> frontend/src/App.jsx Written by Eugene Au

```
import { Navigate, 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";
import Challenge from "./pages/challenge";
import EditProfile from "./pages/editProfile";
import ChangeIcon from "./pages/changeIcon";
import TermsConditions from "./pages/terms-pages/termsConditions";
import PrivacyPolicy from "./pages/terms-pages/privacyPolicy";
import Admin from "./pages/admin";

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 />} />
        <Route path="/challenge" element={<Challenge />} />
        <Route path="/editProfile" element={<EditProfile />} />
        <Route path="/changeIcon" element={<ChangeIcon />} />
        <Route path="/terms-and-conditions" element={<TermsConditions/>} />
        <Route path="/privacy-policy" element={<PrivacyPolicy/>} />
        <Route path="/admin" element={<Admin />} />
      </Routes>
      <Footer />
    </>
  );
}
export default App;
```

styles

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;
}

.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%);
  }
}
```

stores

adminStore.ts

> frontend/src/stores/adminStore.ts Written Adam George

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

type adminStore = {
    isAdmin: boolean;
    setAdmin: (admin: boolean) => void;
}

export const useAdminStore = create<adminStore>((set) => ({
    isAdmin: false,
    setAdmin: (admin) => set({isAdmin: admin})
}));
```

geolocationStore.ts

> frontend/src/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/stores/pinStore.ts Written by Adam George

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

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]})),
}));
```

pages

admin.jsx

> frontend/src/pages/admin.jsx Written Adam George

```
import { useEffect, useState } from 'react';
import { Navigate } from 'react-router-dom';
import axios from 'axios';
import Cookies from 'universal-cookie';
import "./stylesheets/admin.css";
import postimg from '../assets/admin/posts.svg';
import usersimg from '../assets/admin/users.svg';
import databaseimg from '../assets/admin/database.svg';
import DashboardCard from '../features/dashboardCard';

const Admin = () => {
    const [admin, setAdmin] = useState(false);
    const [loading, setLoading] = useState(true);
    const [page, setPage] = useState('posts');
    const [posts, setPosts] = useState(undefined);
    const [users, setUsers] = useState(undefined);
    const [postView, setPostView] = useState(false);
    const [activePost, setActivePost] = useState(undefined);
    const cookies = new Cookies();
    const token = cookies.get('token');


    const checkSuperUser = async () => {
        try {
            const response = await
axios.get('http://localhost:8000/api/isSuperUser/', {
                headers: {
                    'Authorization': `Token ${token}`
                }
            });
            setAdmin(response.data.is_superuser);
        } catch (error) {
            console.log(error);
        }
    };

    const getPosts = async () => {
```

```
        try {
          const response = await axios.get(
            "http://127.0.0.1:8000/api/getAllPosts/", {
                headers: {
                    'Authorization': `Token ${token}`
                }
            });
          return response.data;
        } catch (error) {
          console.error(error);
        } finally {
            setLoading(false);
        }
    };

    const getUsers = async () => {
        try {
          const response = await axios.get(
            "http://127.0.0.1:8000/api/getAllUsers/", {
                headers: {
                    'Authorization': `Token ${token}`
                }
            });
          return response.data;
        } catch (error) {
          console.error(error);
        } finally {
            setLoading(false);
        }
    };

    const deletePost = async (id) => {
        try {
            const response = await axios.get(
              `http://127.0.0.1:8000/api/deletePost/${id}`, {
                headers: {
                    'Authorization': `Token ${token}`
                }
              });
            alert(response.data);
          } catch (error) {
            console.error(error);
          }
    };

    const banAuthor = async (id) => {
        try {
            const response = await axios.get(
              `http://127.0.0.1:8000/api/deleteUser/${id}`, {
                headers: {
                    'Authorization': `Token ${token}`
                }
              });
            alert(response.data);
```

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

        useEffect(() => {
            checkSuperUser();
            getPosts().then((data) => {
                setPosts(data.map(post => ({ ...post, open: false })).sort((a, b) =>
new Date(b.datetime) - new Date(a.datetime)));
            });
            getUsers().then((data) => {
                console.log(data);
                setUsers(data);
            });
        }, []);

        useEffect(() => {
            console.log(activePost);
        }, [activePost]);

        function formatDate(dateString) {
            const options = { hour: '2-digit', minute: '2-digit', day: '2-digit',
month: '2-digit', year: 'numeric' };
            const date = new Date(dateString);
            const formattedDate = `${date.toLocaleTimeString('en-GB', options)}`;

            return formattedDate;
        }

        const handleDashboardPost = (id) => {
            setActivePost(posts.find(post => post.id === id));
            setPostView(true);
        };

        const handlePageChange = (page) => {
            setPage(page);
        };

        if (loading) {
            return <div>Loading...</div>;
        } else {
            return (
                <div id="admin-dashboard">
                    <h1 className='dashboard-title'>Admin Dashboard</h1>
                        <div id="dashboard-navbar">
                            <button type='button' onClick={() =>
handlePageChange('posts')} className={page == 'posts' ? "active-button dashboard-
navbar-buttons" : "dashboard-navbar-buttons"}><img src={postimg}>
</img>Posts</button>
                            <button type='button' onClick={() =>
handlePageChange('users')} className={page == 'users' ? "active-button dashboard-
navbar-buttons" : "dashboard-navbar-buttons"}><img src={usersimg}>
</img>Users</button>
```

```
                    </div>
                    {page == 'posts'  && <div id="dashboard-post-list">
                        {postView &&
                        <DashboardCard
                            id = {activePost?.id}
                            username={activePost?.username}
                            isSuperUser={activePost?.is_superuser}
                            image={activePost?.image}
                            caption={activePost?.caption}
                            datetime={activePost?.datetime}
                            location={activePost?.position.location}
                            userid={activePost?.userid}
                            setPostView={setPostView}
                            deletePost={deletePost}
                            deleteUser={banAuthor} />
                        }
                        <div id="dashboard-post-inner">
                            {!loading && posts.map((post) => {
                                return (
                                    <div className="dashboard-post" key={post.id}
onClick={() => handleDashboardPost(post.id)}>
                                    <img className="dashboard-post-image" src=
{"http://127.0.0.1:8000" + post.image} alt={post.caption} />
                                    <div className="dashboard-post-date">
                                        {formatDate(post.datetime)}
                                    </div>
                                </div>
                                );
                            })}
                        </div>
                    </div>}
                    {page == 'users'  && <div id="dashboard-user-list">
                        <table>
                            <thead>
                                <tr>
                                <th>Id</th>
                                <th>Picture</th>
                                <th>Username</th>
                                <th>Bio</th>
                                <th>Twitter</th>
                                <th>Instagram</th>
                                <th>Youtube</th>
                                <th>Action</th>
                                </tr>
                            </thead>
                            <tbody>
                                {loading ? (<tr><td> Loading ... </td></tr>) :
                                    users.map((user) => (
                                        <tr key={user.id}>
                                            <td>{user.id}</td>
                                            <td>{user.profilePicture}</td>
                                            <td>{user.username}</td>
                                            <td>{user.bio}</td>
                                            <td>{user.twitter}</td>
```

```
                                            <td>{user.instagram}</td>
                                            <td>{user.youtube}</td>
                                            <td>
                                                <button type='button' disabled=
{user.is_superuser} onClick={() => banAuthor(user.id)} className="delete-
button">Ban User</button>
                                            </td>
                                        </tr>
                                    ))
                                }
                            </tbody>
                        </table>
                    </div>}
                </div>
            );
        }
};

export default Admin;
```

capture.jsx

> frontend/src/pages/capture.jsx Written by Eugene Au, 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";
import Polaroid from "../features/polaroid";

// function for set cookies
const cookies = new Cookies();
axios.defaults.withCredentials = true;

// the page
function Capture() {

  // check if the user have logged in if so capture
  useEffect(() => {
    async function check() {
      await CheckLogin();
      capture();
    }
```

```
    check();
  }, []);

  // Hook to redirect user programmatically.
  const navigate = useNavigate();

  // State management for geolocation and location tags using zustand stores.
  const position = usePositionStore(state => state.position);
  const setPosition = usePositionStore(state => state.setPosition);
  const locationTag = useGeoTagStore(state => state.geoTag);
  const setLocationTag = useGeoTagStore(state => state.setGeoTag);


  // Local state management for UI interactions and data handling.
  const [lastPosition, setLastPosition] = useState({ lat: 0, lng: 0 });
  const inputRef = useRef(null);
  const [previewImg, setPreviewImg] = useState("");
  const [file, setFile] = useState(null);
  const [isLoading, setIsLoading] = useState(false);
  const [caption, setCaption] = useState("");
  const [captionError, setCaptionError] = useState("");

  // Handle caption input changes and enforce character limit.
  const handleChange = (e) => {
    const value = e.target.value;
    if (value.length > 255) {
      setCaptionError("Caption cannot exceed 255 characters.");
    } else if (value.length >= 200) {
      setCaptionError(`Approaching limit (${value.length}/255)`);
    } else {
      setCaptionError("");
    }
    setCaption(value);
  };

  // On component mount, check login status and initialize capture functionality.
  useEffect(() => {
    if (navigator.geolocation) {
      navigator.geolocation.watchPosition((position) => {
        setPosition(position.coords.latitude, position.coords.longitude);
      });
    }
  }, []);

  // Update location tag based on position change with a certain threshold.
  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]);

  // Handle form submission for creating a new post.
```

```
    const handleSubmit = async (e) => {
      e.preventDefault();

      // The caption should not be more than 255 charaters
      if (caption.length > 255) {
        return;
      }
      setIsLoading(true);

      // post the geolocation data
      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();
        const imageFile = document.getElementById("fileInput").files[0];
        if (imageFile) {
          formData.append("image", imageFile);
        } else {
          alert("No image file selected");
          setIsLoading(false);
          return;
        }

        formData.append("caption", caption);
        formData.append("geolocID", geolocID);

        // post the post data
        try {
          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')}`,
              },
            }
          );

          navigate("/");
        } catch (error) {
          handleNetworkError(error);
        }
      } catch (error) {
        handleNetworkError(error);
      }
```

```
    setIsLoading(false);
  };

  // called when there's a network error
  const handleNetworkError = (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);
    } else if (error.request) {
      // The request was made but no response was received, likely a network error
      alert("Cannot access the backend. Please check your network connection.");
    } else {
      // Something happened in setting up the request that triggered an Error
      alert("An error occurred while creating the post");
    }
  };


  // open the user's camera
  const capture = () => {
    inputRef.current.click();
  };

  // get the captured file
  const handleCapture = (e) => {
    const file = e.target.files[0];
    if (file) {
      setPreviewImg(URL.createObjectURL(file));
      setFile(file);
    }
  };

  return (
    <>
    {/* the loading screen for posting */}
      <LoadingScreen active={isLoading} />
      <div id="capturePage" className="page active">
        <input
          type="file"
          accept="image/*"
          capture="environment"
          id="fileInput"
          ref={inputRef}
          onChange={handleCapture}
          style={{ display: "none" }}
        />
        <div id="displayCapture">
          <div id="preview-wrapper">
            <div id="preview">
              <div id="spacer">
                <div id="contents">
                  {previewImg ? (
```

```
                    <Polaroid
                      src={previewImg}
                      caption={caption}
                    />
                  ) : (
                    <div
                      id="previewPlaceholder"
                      onClick={
                        () => {
                          capture();
                        }
                      }
                    >
                      <p>Tap to take a picture</p>
                    </div>
                  )}
                  <form onSubmit={handleSubmit}>
                    <div id="form">
                      <input
                        name="caption"
                        type="caption"
                        placeholder="Post caption"
                        value={caption}
                        onChange={handleChange}
                      />
                      {captionError &&
                        <div style={{ width: "100%", textAlign: "right" }}>
                          <div style={{ color: caption.length >= 250 ? "red" :
"orange" }}>{captionError}</div></div>}
                    </div>
                    <div id="previewButtons">
                      <div className="retake element">
                        <img
                          src={Reset}
                          width={"15px"}
                          height={"15px"}
                          onClick={
                            () => {
                              capture();
                            }
                          }
                        />
                      </div>
                      <div className="location element">
                        <img src={Location} />
                        {locationTag}
                      </div>
                      <button
                        className="share element"
                        style={{ height: "100%" }}
                      >
                        <img src={Send} />
                      </button>
                    </div>
```

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

export default Capture;
```

challenge.jsx

> frontend/src/pages/challenge.jsx Written by Eugene Au

```
import './stylesheets/challenge.css';
import Coin from '../assets/challenge/coin.png';
import Cat from '../assets/store/Napoleon.png';

import CheckLogin from '../features/CheckLogin';
import { useEffect, useState } from 'react';
import axios from 'axios';
import Cookies from 'universal-cookie';
import ErrorBox from '../features/ErrorBox';
import usericon from '../assets/header/user-icon.jpg'

const cookies = new Cookies();

function Challenge() {

    // Local state management for UI interactions and data handling.
    const [avatars, setAvatars] = useState([]);
    const [challenges, setChallenges] = useState([]);
    const [token, setToken] = useState(cookies.get('token'))

    // check if the user have logged in and setup everyting
    useEffect(() => {

        // check if the user has logged in and get the token
        CheckLogin();

        // get all the avatars
        const getAvatars = async () => {
            const response = await axios.get(
                "http://127.0.0.1:8000/api/getAvatars/"
                , {
                    headers: {
                        "Content-Type": "application/json",
                        "Authorization": `Token ${token}`,
```

```
                    }
                }
            )
            return response.data
        }
        getAvatars().then((avatars) => {
            setAvatars(avatars);
        });

        // get all the challenges
        const getChallenges = async () => {
            const response = await axios.get(
                "http://127.0.0.1:8000/api/getChallenges/"
                , {
                    headers: {
                        "Content-Type": "application/json",
                        "Authorization": `Token ${token}`,
                    }
                }
            )
            return response.data
        }
        getChallenges().then((challenges) => {
            setChallenges(challenges);
        });
    }, []);

    // function to purchase a avatar given the avatar name
    async function purchase(StickersName) {

        try {
            const response = await axios.post(
                "http://127.0.0.1:8000/api/purchase/"
                , {
                    "sticker": StickersName
                },
                {
                    headers: {
                        "Content-Type": "application/json",
                        "Authorization": `Token ${token}`,
                    },
                }
            )

            // reload the page after puchase
            location.reload();
        } catch (error) {
            if (error.response.data.Message) {
                alert(error.response.data.Message)
            }
            else {
                alert("Internal server error")
            }
        }
```

```
    }


    return (
        <>
            {/* display if theres an error(unused) */}
            <ErrorBox />
            <div id='challenge'>
                <div id='spacer'>
                    <div className='title'>
                        Challenges
                    </div>
                    <div id='main'>
                        <div id='daily' className='challenge-box-with-title'>
                            Daily
                            <div className='challenge-box'>
                                <div className='challenge-description'>
                                    <div className='content'>
                                        <div className='content-title'>
                                            {challenges.DailyChallenge}
                                        </div>
                                        <div className='progress-bar'>
                                            <div className='progress'>
                                                <div className='progress-bar-fill'
style={{ width: eval(challenges.Daily) * 100 + "%" }}></div>
                                            </div>
                                            <div>{challenges.Daily}</div>
                                        </div>
                                    </div>
                                </div>
                                <div className='reward-amount'>
                                    {challenges.DailyCoinsRewarded}
                                    <img src={Coin} alt='coin' />

                                </div>
                            </div>
                        </div>
                        <div id='milestones' className='challenge-box-with-title'>
                            Milestones
                            <div className='challenge-box'>
                                <div className='challenge-description'>
                                    <div className='content'>
                                        <div className='content-title'>
                                            {challenges.Milestone1Challenge}
                                        </div>
                                        <div className='progress-bar'>
                                            <div className='progress' >
                                                <div className='progress-bar-fill'
style={{ width: eval(challenges.Milestone1) * 100 + "%" }}></div>
                                            </div>
                                            <div>{challenges.Milestone1}</div>
                                        </div>
                                    </div>
                                </div>
```

```
                                <div className='reward-amount'>
                                    {challenges.Milestone1CoinsRewarded}
                                    <img src={Coin} alt='coin' />
                                </div>
                            </div>
                            <div className='challenge-box'>
                                <div className='challenge-description'>
                                    <div className='content'>
                                        <div className='content-title'>
                                            {challenges.Milestone2Challenge}
                                        </div>
                                        <div className='progress-bar'>
                                            <div className='progress' >
                                                <div className='progress-bar-fill'
style={{ width: eval(challenges.Milestone2) * 100 + "%" }}></div>
                                            </div>
                                            <div>{challenges.Milestone2}</div>
                                        </div>
                                    </div>
                                </div>
                                <div className='reward-amount'>
                                    {challenges.Milestone2CoinsRewarded}
                                    <img src={Coin} alt='coin' />
                                </div>
                            </div>
                        </div>
                        <div id='shop' className='challenge-box-with-title'>
                            Shop
                            <div id='shop-box'>
                                {/* dummy item to enforce grid */}
                                <div style={{ width: '85px' }} />
                                <div style={{ width: '85px' }} />
                                <div style={{ width: '85px' }} />
                                <div id={"forth"} style={{ width: '85px' }} />
                                {avatars.map((avatar) => (
                                    <div className='shop-item' key={avatar.name}
onClick={() => { purchase(avatar.name) }}>
                                        <img src={avatar.path === "NULL" ?
usericon : avatar.path} alt='cat' width={"85px"} height={"85px"} style={{ border:
"none", borderRadius: "100%" }} />
                                        {avatar.name}
                                        <div className='cost'>
                                            {avatar.price}
                                            <img src={Coin} alt='coin' width=
{"15px"} height={"15px"} />
                                        </div>
                                    </div>
                                ))}
                            </div>
                        </div>
                    </div>
                </div>

            </div>
```

```
        </>
    )
}

export default Challenge;
```

editProfile.jsx

> frontend/src/pages/editProfile.jsx Written by Eugene Au

```
import './stylesheets/editProfile.css';

import { Link } from 'react-router-dom';
import { useEffect, useState } from 'react';
import CheckLogin from '../features/CheckLogin';
import usericon from "../assets/header/user-icon.jpg";
import axios from 'axios';
import Cookies from 'universal-cookie';

const cookies = new Cookies()

function editProfile() {

    // the local variables
    const [user, setUser] = useState({})
    const [profilePicture, setProfilePicture] = useState("")
    const [profileData, setProfileData] = useState({
        bio: '',
        youtube: '',
        instagram: '',
        twitter: ''
    })

    // get the user's icon and get their profile
    useEffect(() => {
        const getIcon = async () => {
            let response = await CheckLogin();
            return response.data
        }
        getIcon().then((user) => {
            setUser(user)
            if (user.profilePicture !== "NULL") {
                setProfilePicture(user.profilePicture)
            } else {
                setProfilePicture(usericon)
            }
            setProfileData({
                bio: user.Bio,
                youtube: user.youtube,
                instagram: user.instagram,
                twitter: user.twitter
```

```
            })
        });
    }, []);

    const handleChange = (e) => {
        setProfileData({ ...profileData, [e.target.name]: e.target.value })
        console.log(profileData)
    }



  const handleSubmit = async (e) => {
    const token = cookies.get('token');
    e.preventDefault();
    try {
      // Update the API URL as per your configuration
      const response = await axios.post(
        "http://127.0.0.1:8000/api/changeBio/"
        , profileData,
        {
          headers: {
            "Content-Type": "application/json",
            "Authorization": `Token ${token}`,
          },
        }
      );
      location.reload()
    } 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 id='editProfile'>
            <div id='main'>
                <div id='spacer'>
                    <div id='content'>
                        <div id='title'>Edit Profile</div>
                        <div id='profilePic'>
                            <img src={profilePicture} alt='profile pic' />
                        </div>
                        <Link to={"/changeIcon"}>
                            <button style={{ width: "auto", fontSize: "16px",
padding: "5px 10px", marginTop: "20px" }}>Change Icon</button>
                        </Link>
                        <form>
                            <div className='field'>
                                <label for='name'>Bio</label>
                                <input type='text' id='bio' name='bio' value=
```

```
{profileData.bio} onChange={handleChange} />
                            </div>
                            <div className='field'>
                                <label for='name'>Youtube</label>
                                <input id='youtube' name='youtube' value=
{profileData.youtube} onChange={handleChange} />
                            </div>
                            <div className='field'>
                                <label for='name'>Instagram</label>
                                <input type='text' id='instagram' name='instagram'
value={profileData.instagram} onChange={handleChange} />
                            </div>
                            <div className='field'>
                                <label for='name'>Twitter</label>
                                <input type='text' id='twitter' name='twitter'
value={profileData.twitter} onChange={handleChange} />
                            </div>
                            <button onClick={handleSubmit}
type='submit'>Save</button>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    );
}

export default editProfile;
```

changeIcon.jsx

> frontend/src/pages/changeIcon.jsx Written by Eugene Au

```
import './stylesheets/changeIcon.css'
import Cat from '../assets/store/Napoleon.png';
import axios from 'axios';
import { useState, useEffect } from 'react';
import Cookies from 'universal-cookie';
import CheckLogin from '../features/CheckLogin';
import usericon from '../assets/header/user-icon.jpg'

const cookies = new Cookies();


function ChangeIcon() {

    const [avatars, setAvatars] = useState([]);
    const [user, setUser] = useState()
    const [profilePicture, setProfilePicture] = useState("")

    // set the user icon and the user data
```

```
    useEffect(() => {
        const getIcon = async () => {
            let response = await CheckLogin();
            return response.data
        }
        getIcon().then((user) => {
            setUser(user)
            if (user.profilePicture !== "NULL") {
                setProfilePicture(user.profilePicture)
            } else {
                setProfilePicture(usericon)
            }
        });
    }, []);

    useEffect(() => {
        const token = cookies.get('token');

        // get all the unlocked avatars
        const getIcons = async () => {
            // get all the locked avatar and set them
            try {
                const response = await axios.get(
                    "http://127.0.0.1:8000/api/getAllAvatars/"
                    ,
                    {
                        headers: {
                            "Content-Type": "application/json",
                            "Authorization": `Token ${token}`,
                        },
                    }
                );
                return response.data;
            } catch (error) {
                console.error("Error occurred:", error);
                if (error.response) {
                    alert("internal server error")
                }
            }
        }
        getIcons().then((data) => {
            setAvatars(data);
        });

    }, []);

    // set the user's icon
    async function setIcon(avatar) {
        const token = cookies.get('token');
        try {
            const response = await axios.post(
                "http://127.0.0.1:8000/api/changeAvatar/",
                {
                    avatar: avatar
```

```
                },
                {
                    headers: {
                        "Content-Type": "application/json",
                        "Authorization": `Token ${token}`,
                    },
                }
            );
            location.reload();
        } catch (error) {
            console.error("Error occurred:", error);
            if (error.response) {
                alert("internal server error")
            } else {
                alert("cannot connect to server")
            }
        }
    }

    return (
        <div id="changeIcon">
            <div id='spacer'>
                <div id='content'>
                    <div id='title'>Change Icon</div>
                    <div id='user-icon'>
                        <img src={profilePicture} alt='profile pic' />
                    </div>
                    <div id='selections'>
                        <div id='title'>Select Icon</div>

                        <div id='selection-box'>
                            {/* dummy item to enforce grid */}
                            <div style={{ width: '85px' }} />
                            <div style={{ width: '85px' }} />
                            <div style={{ width: '85px' }} />
                            <div id={"forth"} style={{ width: '85px' }} />
                            {
                                avatars.map((avatar) => (
                                    <div className='selection-item' key=
{avatar.name} onClick={() => { setIcon(avatar.name) }}>
                                        <img src={avatar.path === "NULL" ?
usericon : avatar.path} alt='cat' width={"85px"} height={"85px"} style={{ border:
"none", borderRadius: "100%" }} />
                                        {avatar.name}
                                    </div>
                                ))
                            }
                        </div>
                    </div>
                </div>
            </div>
        </div>

    )
```

```
}

export default ChangeIcon;
```

feed.jsx

> frontend/src/pages/feed.jsx Written by Eugene Au , Adam Geroge

```
import React from "react";
import "./stylesheets/feed.css";
import { useState, useEffect } from "react";
import PostView from "../features/PostView";
import Polaroid from "../features/polaroid";
import axios from "axios";
import Cookies from "universal-cookie";
import CheckLogin from "../features/CheckLogin";
import InitMap from "../features/InitMap";
import { Link } from "react-router-dom";



const cookies = new Cookies();


function FeedPage() {

  // check if the user is logged in
  useEffect(() => {
    CheckLogin();
  }, []);

  const [activePost, setActive] = useState({});
  const [noPost, setNoPost] = useState(false);
  const [columns, setColumns] = useState([]);

  const [progress, setProgress] = useState(0);

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


    const token = cookies.get('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 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("Cannot connect to the server");
    }
  };

  useEffect(() => {
    // Function to load an image and update its height in the state
    function getImageRatio(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 / img.width);
        };

        // 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();
      if (postList.length === 0) {
        setNoPost(true);
      }

      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 imageRatio = await getImageRatio(image);

        // if the right column is shorter, add the image to the right column
        if (heightDifference < 0) {
          rightPosts[i] = postList[i];
          heightDifference += imageRatio * 1.1; // 1.1% for the padding of the
polaroid
        } else {
          leftPosts[i] = postList[i];
          heightDifference -= imageRatio * 1.1; // 1.1% for the padding of the
polaroid
        }
        setColumns([leftPosts, rightPosts]);
        setProgress((i / postList.length) * 100);


      }
    };

    processImages();

  }, []);

  return (
    <>
      {/* the loading screen */}
      {/* {loadingImage && <InitMap progress={progress} />} */}
      {/* the absolute position post view */}
      <PostView
        isActive={Object.keys(activePost).length !== 0}
        image={activePost["image"]}
        leaveFunction={() => {
          setActive({});
        }}
        caption={activePost["caption"]}
        location={"Forum"}
        showBottomBar={false}
      />

      {/* prompt the user to collect some posts if there is no post */}
      {noPost && (
        <div id="no-post" style={{ position: "fixed", zIndex: "9", width: "100%",
height: "100vh" }}>
          <div style={{ position: "absolute", transform: "translate(-50%,-50%)",
top: "50%", left: "50%", minWidth: "250px" }}>
            <div style={{ padding: "70px 0 ", color: "black", textAlign: "center",
fontWeight: "700" }}>
              <div style={{ marginBottom: "10px" }}>You have no collected post yet
              </div>
              <Link to="/">
```

```
                <button id="button" style={{
                  width: "100%",
                  padding: "10px",
                  border: "none",
                  borderRadius: "100px",
                  background: "var(--primary)",
                  fontWeight: "700",
                  fontSize: "16px",
                  fontFamily: "Outfit",
                }}>Go to collect</button>
              </Link>
            </div>

          </div >
        </div >
      )
      }

      {/* the feed */}
      {/* if there is no post or the post view is active, blur the feed */}
      <div id="feed" className={Object.keys(activePost).length !== 0}>
        <div id="padding">
          <div id="daily-feed">
            <div id="grid-wrapper">
              {noPost ? (
                <>
                  <div className={"image-grid "}>
                    <div className="polaroid skeleton shadow">
                      <div className="padding skeleton">
                        <div className="image skeleton" style={{ aspectRatio: 4 /
5 }}></div>
                      </div>
                    </div>
                    <div className="polaroid skeleton shadow">
                      <div className="padding skeleton">
                        <div className="image skeleton"></div>
                      </div>
                    </div>
                  </div>

                  <div className={"image-grid "}>
                    <div className="polaroid skeleton shadow">
                      <div className="padding skeleton">
                        <div className="image skeleton"></div>
                      </div>
                    </div>
                    <div className="polaroid skeleton shadow">
                      <div className="padding skeleton">
                        <div className="image skeleton" style={{ aspectRatio: 4 /
5 }}></div>
                      </div>
                    </div>
                  </div>
                </>
```

```
              ) : (
                // map the 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>
      </div >
    </>
  );
}

export default FeedPage;
```

login.jsx

> frontend/src/pages/login.jsx Written by Eugene Au, Adam George

```
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();

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

```
    // the user details
    const [userData, setUserData] = useState({
      username: "",
      password: "",
    });

    // get text from the forms to the data
    const handleChange = (e) => {
      setUserData({
        ...userData,
        [e.target.name]: e.target.value,
      });
    };

    // submit the form
    const hansleSubmit = async (e) => {
      e.preventDefault();
      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("Internal server error. Please try again later.");
          }
        } else {
          console.log(errors);
          alert("Cannot connect to the server");
        }
      }


    }

    return (
      <div id="displayLogin">
        <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";
import question from "../assets/map/question.svg";

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
  useEffect(() => {
    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;
  }

  const discoverPins = (lat, lng) => {
    const minRadius = 0.0005; // Minimum radius of discovery (about 35m from the
position)
    const maxRadius = 0.0025; // Maximum radius of discovery (about 175m from the
position)

    const discoverPins = 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 > minRadius && distance < maxRadius;
      });
      return discoverPins;
    }


    // 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"]}
```

```
        />
        <div id="map-content">
          {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>
              );
            })}
            {discoverPins(position.lat, position.lng).map((pin) => {
              const color = `var(--${mood})`;
              return (
                <AdvancedMarker
                  key={pin.id}
                  position={pin.position}
                >
                  <img src={question} />
                </AdvancedMarker>
              );
            })}
          </Map>
        </div>
      </APIProvider>}
      {showMoodPrompt && <MoodPrompt onClickFunction={handleMoodPrompt} />}
      <div className="bottom-prompt">
        <div className="bottom-prompt-wrapper">
          <img src={Location} />{locationTag}
        </div>
      </div>
    </div>

  </>
  );
}

export default MapPage;
```

pageNoFound.jsx

> frontend/src/pages/pageNoFound.jsx Written by Adam George , Eugene Au

```
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";
import ytIcon from "../assets/profilepage/YouTube.svg";
import instaIcon from "../assets/profilepage/Instagram.svg";
import twitterIcon from "../assets/profilepage/twitter.svg";
import CheckLogin from "../features/CheckLogin";
import { useState, useEffect } from "react";

function ProfilePage() {

  const [user, setUser] = useState({});
  async function setUserName() {
    let response = await CheckLogin()
    setUser(response.data)
  }
  useEffect(() => {
    setUserName();
  }, []);

  console.log(user)

  return (

    <div id="displayProfile">
      <div id="profile-wrapper">
        <div id="profile">
          <div id="spacer">
            <div id="title">{user.username}</div>
            <div id="user-icon">
              <img src={user.profilePicture === "NULL" ? userIcon :
user.profilePicture} alt="user icon" width={"100%"} />
            </div>
            <div id="bio">
              {user.Bio}
              <hr></hr>
            </div>
            <div id="socials">
              <a id="youtube" className="social-icon" href={user.youtube}>
                <img src={ytIcon} alt="YouTube" width={"17px"} height={"17px"} />
                Youtube
              </a>
              <a href={user.instagram} id="instagram" className="social-icon">
```

```
                <img src={instaIcon} alt="YouTube" width={"17px"} height={"17px"}
/>
                Instagram
              </a>
              <a id="twitter" className="social-icon" href={user.twitter}>
                <img src={twitterIcon} alt="YouTube" width={"17px"} height=
{"17px"} />
                Twitter
              </a>
            </div>
          </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, Link } 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();

  // local state data
  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);

  // get the text from the form to the
  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,
            });
          }
          console.log(error.response.data);
          if (error.response.data["error messages"]) {
            e = error.response.data["error messages"]
            setErrors({
              ...errors,
              username: error.response.data.username,
            });
          }
          if (error.response.data.email) {
            setErrors({
              ...errors,
              email: error.response.data.email,
            });
          }
          if (error.response.data["Password error"]) {
            setErrors({
              ...errors,
              password: error.response.data["Password error"],
            });
          }
        } else {
          // The request was made but no response was received or other errors
occurred
          console.error("Error:", error.message);
          alert("Cannot connect to the server");
        }
        // remove the loading screen
        setIsLoading(false);
```

```
        return;
        // Handle error (e.g., show error message)
      }
    };

    return (
      <>
        <LoadingScreen active={isLoading} />
        <div id="displayRegister">
          <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"
                      />
                      {errors.username && <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"
                      />
                      {errors.email && <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"
                      />
                      {errors.password && <label className="error">{errors.password}
</label>}
                    </div>
```

```
                  <div className="field">
                    <input
                      className="text input"
                      type="password"
                      placeholder="Confirm Password"
                      name="confirmPassword"
                      onChange={handleChange}
                      required
                    />
                    {errors.confirmPassword && <label className="error">
{errors.confirmPassword}</label>}
                  </div>
                  <div className="field">
                    <label className="checkbox">
                      <input
                        type="checkbox"
                        onChange={handleCheckBoxChange}
                        checked={isChecked}
                      />
                      <span className="checkmark"></span>
                      <div className="checkmark-label">
                      I agree to Postpal's{' '}
                      <Link to={"/terms-and-conditions"} className="inline-
link">Terms & Conditions</Link>{' '} and {' '}
                      <Link to={"/privacy-policy"} className="inline-link">Privacy
Policy</Link>
                      <br />
                      </div>
                    </label>
                    {errors.checkbox &&
                      <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, { useState } from "react";
import axios from "axios";
```

```
import Cookies from "universal-cookie";
import './stylesheets/test.css';

const cookies = new Cookies();

function Test() {
  const token = cookies.get('token');
  console.log(token);

  const [form, setForm] = useState({
    userId: '', // You need to capture the user ID somehow, adjust as necessary
  });

  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/getUser/"
        , {
          bio: "I love django :3",
          youtube: "https://www.youtube.com/",
          twitter: "changedTwitter",
          instagram: "changedInstagram",
        },
        {
          headers: {
            "Content-Type": "application/json",
            "Authorization": `Token ${token}`,
          },
        }
      );
      console.log(response.data); // Assuming response.data contains the username
    } 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);
      }
    }
  };

  // Update the state to capture the user ID from an input field
  const handleUserIdChange = (e) => {
    setForm({ ...form, userId: e.target.value });
  };

  return (
    <div className="test">
      <form onSubmit={handleSubmit}>
        <input
          type="text"
          placeholder="User ID"
```

```
            value={form.userId}
            onChange={handleUserIdChange}
          />
          <button type="submit">Submit</button>
        </form>
      </div>
    );
}

export default Test;
```

terms-pages

privacyPolicy.jsx

> frontend/src/pages/terms-pages/privacyPolicy.jsx Written by Adam George

```
import "../stylesheets/termsConditions.css";
import caret from "../../assets/terms/back-caret.svg";
import { Link } from "react-router-dom";

const PrivacyPolicy = () => {
    return (
        <>
            <div className="terms-conditions">
            <Link to={"/register"}>
                <div className="go-back">
                    <img src={caret} height="18px"></img>
                    Go back
                </div>
            </Link>
            <h1>Privacy Policy for Post-i-tivity</h1>
            <p>Effective date: 22/02/2024<br/>
            ExceptionHandler ("us", "we", or "our") operates the Post-i-tivity
mobile application (here in after referred to as the "Service").
            This page informs you of our policies regarding the collection, use,
and disclosure of personal data when you use our Service and the choices you have
associated with that data.
            We use your data to provide and improve the Service. By using the
Service, you agree to the collection and use of information in accordance with
this policy.
            </p>
            <h2>1. Information Collection and Use</h2>
            <p>We collect several different types of information for various
purposes to provide and improve our Service to you.</p>
            <h3>Types of Data Collected</h3>
            <h4>Personal Data</h4>
            <p>While using our Service, we may ask you to provide us with certain
personally identifiable information that can be used to contact or identify you
("Personal Data"). Personally identifiable information may include, but is not
limited to:</p>
```

```
            <ul>
                <li>Email address</li>
                <li>Location data</li>
                <li>Cookies and Usage Data</li>
            </ul>
            <p>When you access the Service by or through a mobile device, we may
collect certain information automatically, including, but not limited to, the type
of mobile device you use, your mobile device's unique ID, the IP address of your
mobile device, your mobile operating system, the type of mobile Internet browser
you use, unique device identifiers and other diagnostic data ("Usage Data").</p>
            <h2>2. Use of Data</h2>
            <p>ExceptionHandler uses the collected data for various purposes:</p>
            <ul>
                <li>To provide and maintain the Service</li>
                <li>To allow you to participate in interactive features of our
Service when you choose to do so</li>
                <li>To provide customer care and support</li>
                <li>To provide analysis or valuable information so that we can
improve the Service</li>
                <li>To monitor the usage of the Service</li>
                <li>To detect, prevent and address technical issues</li>
            </ul>
            <h2>3. Transfer of Data</h2>
            <p>
                Your information, including Personal Data, may be transferred to —
and maintained on — computers located outside of your state, province, country or
other governmental jurisdiction where the data protection laws may differ than
those from your jurisdiction.<br/>
                Your consent to this Privacy Policy followed by your submission of
such information represents your agreement to that transfer.<br/>
                ExceptionHandler will take all steps reasonably necessary to
ensure that your data is treated securely and in accordance with this Privacy
Policy and no transfer of your Personal Data will take place to an organization or
a country unless there are adequate controls in place including the security of
your data and other personal information.

            </p>
            <h2>4. Disclosure of Data</h2>
            <h4>Legal Requirements</h4>
            <p>ExceptionHandler may disclose your Personal Data in the good faith
belief that such action is necessary to:</p>
            <ul>
                <li>To comply with a legal obligation</li>
                <li>To protect and defend the rights or property of Exception
Handler</li>
                <li>To prevent or investigate possible wrongdoing in connection
with the Service</li>
                <li>To protect the personal safety of users of the Service or the
public</li>
                <li>To protect against legal liability</li>
            </ul>
            <h2>5. Security of Your Information</h2>
            <p>We value your trust in providing us with your Personal Information,
thus we strive to use commercially acceptable means of protecting it. But remember
```

```
that no method of transmission over the internet, or method of electronic storage
is 100% secure and reliable, and we cannot guarantee its absolute security.</p>
            <h2>6. Changes to This Privacy Policy</h2>
            <p>We may update our Privacy Policy from time to time. Thus, you are
advised to review this page periodically for any changes. These changes are
effective immediately after they are posted on this page.</p>

            <h2>Contact Us</h2>
            <p>If you have any questions about these Terms, please contact us at
rb901@exeter.ac.uk.</p>
            </div>
        </>
    );
}

export default PrivacyPolicy;
```

termsConditions.jsx

> frontend/src/pages/terms-pages/termsConditions.jsx Written by Adam George

```
import "../stylesheets/termsConditions.css";
import caret from "../../assets/terms/back-caret.svg";
import { Link } from "react-router-dom";

const TermsConditions = () => {
    return (
        <>
            <div className="terms-conditions">
                <Link to={"/register"}>
                    <div className="go-back">
                        <img src={caret} height="18px"></img>
                        Go back
                    </div>
                </Link>
                <h1>Terms and Conditions for Post-i-tivity</h1>
                <p>Last updated: 22/02/2024 <br/>
                    Welcome to Post-i-tivity! These Terms and Conditions outline
the rules and regulations for the use of  Post-i-tivity's mobile application and
website (if applicable).
                    By accessing this app, we assume you accept these terms and
conditions. Do not continue to use Post-i-tivity if you do not agree to take all
of the terms and conditions stated on this page.
                </p>
                <h2>
                    1. Definitions
                </h2>
                <p>
                    For the purposes of these Terms and Conditions:
                </p>
                <ul>
```

```
                    <li>"App" refers to Post-i-tivity.</li>
                    <li>"You" means the individual accessing the App, or the
company, or other legal entity on behalf of which such individual is accessing or
using the App.</li>
                    <li>"Company" (referred to as either "the Company", "We", "Us"
or "Our" in this Agreement) refers to Exception Handler</li>
                    <li>"Content" means any audio, video, text, images, or other
material you choose to display on this App.</li>
                    <li>"Terms and Conditions" (also referred as "Terms") mean
these Terms and Conditions that form the entire agreement between You and the
Company regarding the use of the App.</li>
                </ul>
                <h2>2. User Accounts</h2>
                <p>When you create an account with us, you must provide us with
information that is accurate, complete, and current at all times. Failure to do so
constitutes a breach of the Terms, which may result in immediate termination of
your account on our App.</p>
                <h2>3. Use of Service</h2>
                <p>You are granted a limited, non-exclusive, revocable license to
access and use the App strictly in accordance with these Terms.</p>
                <h2>4. Intellectual Property</h2>
                <p>The App and its original content (excluding Content provided by
You or other users), features, and functionality are and will remain the exclusive
property of Exception Handler and its licensors.</p>
                <h2>5. User Content</h2>
                <p>You are responsible for the Content that you post on or through
the App, including its legality, reliability, and appropriateness.</p>
                <h2>6. Privacy Policy</h2>
                <p>Please refer to our Privacy Policy for information on how we
collect, use, and share your data.</p>
                <h2>7. Termination</h2>
                <p>We may terminate or suspend your account immediately, without
prior notice or liability, for any reason whatsoever, including without limitation
if you breach the Terms.</p>
                <h2>8. Limitation of Liability</h2>
                <p>In no event shall Exception Handler, nor any of its directors,
employees, partners, agents, suppliers, or affiliates, be liable for any indirect,
incidental, special, consequential or punitive damages, including without
limitation, loss of profits, data, use, goodwill, or other intangible losses,
resulting from your access to or use of or inability to access or use the App.</p>
                <h2>9. Governing Law</h2>
                <p>These Terms shall be governed and construed in accordance with
the laws of the United Kingdom, without regard to its conflict of law provisions.
</p>
                <h2>10. Changes to Terms</h2>
                <p>We reserve the right, at our sole discretion, to modify or
replace these Terms at any time.</p>
                <h2>11. Contact Us</h2>
                <p>If you have any questions about these Terms, please contact us
at rb901@exeter.ac.uk.</p>
            </div>
        </>
    );
}
```

```
export default TermsConditions;
```

stylesheets

admin.css

> frontend/src/pages/stylesheets/admin.css Written by Adam George

```
#admin-dashboard {
    margin-top: 60px;
}

.dashboard-title {
  font-size: 1.5em;
  margin-top: 2em;
  margin-left: 1em;
  color: black;
}

#dashboard-post-list {
  margin-top: 2em;
  margin-left: 1em;
  margin-right: 1em;
  background-color: white;
  display: flex;
  gap: 2px;
}

.dashboard-post {
  display: flex;
  flex-direction: column;
  height: 120px;
  padding: 5px;
  border: 1px solid #e0e0e0;
  transition: transform 0.2s ease-in-out;
}

.dashboard-post:hover {
  cursor: pointer;
  transform: scale(0.95) rotate(2deg);
}

.dashboard-post-image {
    width: auto;
    height: 90%;
}

.dashboard-post-date {
    margin-top: 2px;
    color:black;
```

```
    font-family: Outfit, sans-serif;
    font-weight: 500;
    font-size: 0.6em;
}

#dashboard-navbar{
  display: flex;
  margin: 0.5em 1em;
  gap: 1em;
}

.dashboard-navbar-buttons{
  font-family: Outfit, sans-serif;
  font-weight: 600;
  font-size: 1em;
  display: flex;
  align-items: center;
  background-color: white;
  color: black;
  border: 2px solid var(--primary);
  border-radius: 50px;
  padding: 0.2em 1em;
  transition: transform 0.2s, background-color 0.2s;
}

.dashboard-navbar-buttons img{
  width: 20px;
  height: 20px;
  margin-right: 0.5em;
}

.dashboard-navbar-buttons:hover {
  background-color: var(--primary);
  transform: scale(1.05);
  cursor: pointer;
}

.active-button{
  background-color: var(--primary);
}

#dashboard-post-inner{
    display: flex;
    margin-left: 1em;
}

#dashboard-user-list{
  display: flex;
  justify-content: center;
}

#dashboard-user-list table{
  border-collapse: collapse;
  width: 95vw;
```

```
  margin-top: 33px;
}
#dashboard-user-list table, #dashboard-user-list th, #dashboard-user-list td {
  border-style: hidden;
  text-align: center;
}
#dashboard-user-list thead, #dashboard-user-list thead tr{
  color: black;
  background-color: var(--primary);
}

#dashboard-user-list thead{
  height: 42px;
  border-radius: 9px;
}
#dashboard-user-list table th:first-child{
  border-radius:9px 0 0 9px;
}
#dashboard-user-list table th:last-child{
  border-radius:0 9px 9px 0;
}
#dashboard-user-list tr:nth-child(even) {
  background-color: #faf5f0;
}
#dashboard-user-list tbody tr{
  color: black;
  height: auto;
}
#dashboard-user-list tbody td{
  font-weight: 400;
  font-size: 16px;
}
#dashboard-user-list th{
  font-weight: 600;
  font-size: 16px;
}

#dashboard-user-list .delete-button {
  background: rgb(248, 70, 70);
  font-family: Outfit, sans-serif;
  font-weight: 500;
  border: none;
  padding: 0.2em 1em;
  border-radius: 50px;
}

#dashboard-user-list .delete-button:hover {
  background: rgb(255, 0, 0);
  cursor: pointer;
}

#dashboard-user-list .delete-button:disabled {
  background: rgb(238, 128, 128);
  cursor: none;
```

```
}
```

capture.css

> frontend/src/pages/stylesheets/capture.css Written by Eugene Au

```
#displayCapture {
  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 {
  width: inherit;
  /* max-height: 100vh; */
  top: 50%;
  width: 100%;
}

#preview #spacer {
  padding: 70px 0;
  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%;
}

#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;

  margin-top: 20px;
}

#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;

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

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

  margin-bottom: 5px;
}
```

challenge.css

> frontend/src/pages/stylesheets/challenge.css Written by Eugene Au

```
#challenge {
  display: flex;
```

```
  flex-direction: column;
  align-items: center;
  height: 100vh;
  color: black;
}

/* for the header and the footer */
#challenge #spacer {
  padding: 70px 0;
  width: 100%;
}

#challenge .title {
  margin-top: 30px;
  font-size: 36px;
  font-weight: 700;
  text-align: center;
}

#challenge #main {
  width: calc(100% - 40px);
  margin-left: auto;
  margin-right: auto;
  min-width: 290px;
  max-width: 500px;
}

.challenge-box-with-title {
  font-size: 22px;
  font-weight: 700;
}

#challenge #daily {
  margin-top: 20px;
}

.challenge-box-with-title .title {
  font-size: 22px;
  font-weight: 700;
}

.challenge-box {
  background-color: #00dca5;
  border: none;
  border-radius: 12px;
  display: flex;
  flex-direction: row;
  margin-top: 10px;
}

.challenge-description {
  font-size: 20px;
  font-weight: 600;
  text-align: center;
```

```
  background-color: #35b39a;
  border: none;
  border-top-left-radius: inherit;
  border-bottom-left-radius: inherit;
  padding: 5px 10px;
  font-size: 16px;
  width: 100%;
}

.reward-amount {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  font-size: 12px;
  font-weight: 600;
  width: 20%;
}

.content-title {
  font-size: 16px;
  text-align: left;
}

.progress-bar {
  width: 100%;
  height: 15px;
  margin-top: 7px;
  border: none;
  border-radius: 100px;
  display: flex;
  align-items: center;
  justify-content: center;
  font-size: 12px;
  margin-bottom: 5px;
}

.progress-bar .progress {
  background-color: #1d9299;
  border: none;
  border-radius: 100px;
  height: 100%;
  width: 100%;
  margin-right: 10px;
}

.progress-bar-fill {
  background-color: #00dca5;
  border: none;
  border-radius: 12px;
  width: 0%;
  height: 100%;
  transition: width 1s;
}
```

```
#milestones {
  margin-top: 20px;
}

#milestones .background {
  padding: 10px;
  background-color: whitesmoke;
  border: none;
  border-radius: 12px;
}

#shop {
  margin-top: 20px;
}

#shop-box {
  margin-top: 20px;
  flex-direction: row;
  justify-content: space-between;
  display: grid;
  justify-items: start;
  align-content: center;
  grid-template-columns: auto auto auto;
  gap: 10px;
}

#shop-box #forth {
  display: none;
}

@media screen and (min-width: 500px) {
  #shop-box {
    grid-template-columns: auto auto auto auto;
  }

  #shop-box #forth {
    display: block;
  }
}

.shop-item {
  font-size: 12px;
  display: flex;
  flex-direction: column;
}

.shop-item .cost {
  display: flex;
  flex-direction: row;
  align-items: center;
}
```

changeIcon.css

> frontend/src/pages/stylesheets/changeIcon.css Written by Eugene Au

```
#changeIcon {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
}

#changeIcon #title {
  font-size: 36px;
  font-weight: 700;
  color: black;
  text-align: center;
}

#changeIcon #spacer {
  width: 100%;
  padding: 70px 0;
}

#changeIcon #content {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  width: calc(100% - 40px);
  margin-left: auto;
  margin-right: auto;
  max-width: 500px;
}

#changeIcon #user-icon {
  width: 100%;
  display: flex;
  align-items: center;
  justify-content: center;
  margin-top: 40px;
}

#changeIcon #user-icon img {
  width: 90px;
  height: 90px;
  border: none;
  border-radius: 90px;
}

#changeIcon #selections {
  width: 100%;
```

```
  margin-top: 35px;
}

#changeIcon #selections #title {
  font-size: 20px;
  font-weight: 700;
  text-align: left;
}

#selection-box {
  margin-top: 20px;
  flex-direction: row;
  justify-content: space-between;
  display: grid;
  justify-items: start;
  align-content: center;
  grid-template-columns: auto auto auto;
  gap: 10px;
  color: black;
  font-size: 12px;
  font-weight: 700;
  text-align: center;
}

#selection-box #forth {
  display: none;
}

@media screen and (min-width: 500px) {
  #selection-box {
    grid-template-columns: auto auto auto auto;
  }

  #selection-box #forth {
    display: block;
  }
}

.selection-item {
  font-size: 12px;
  display: flex;
  flex-direction: column;
}

.selection-item .cost {
  display: flex;
  flex-direction: row;
  align-items: center;
}
```

changeIcon.css

frontend/src/pages/stylesheets/changeIcon.css Written by Eugene Au

```css
#changeIcon {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
}

#changeIcon #title {
  font-size: 36px;
  font-weight: 700;
  color: black;
  text-align: center;
}

#changeIcon #spacer {
  width: 100%;
  padding: 70px 0;
}

#changeIcon #content {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  width: calc(100% - 40px);
  margin-left: auto;
  margin-right: auto;
  max-width: 500px;
}

#changeIcon #user-icon {
  width: 100%;
  display: flex;
  align-items: center;
  justify-content: center;
  margin-top: 40px;
}

#changeIcon #user-icon img {
  width: 90px;
  height: 90px;
  border: none;
  border-radius: 90px;
}

#changeIcon #selections {
  width: 100%;
  margin-top: 35px;
}
```

```
#changeIcon #selections #title {
  font-size: 20px;
  font-weight: 700;
  text-align: left;
}

#selection-box {
  margin-top: 20px;
  flex-direction: row;
  justify-content: space-between;
  display: grid;
  justify-items: start;
  align-content: center;
  grid-template-columns: auto auto auto;
  gap: 10px;
  color: black;
  font-size: 12px;
  font-weight: 700;
  text-align: center;
}

#selection-box #forth {
  display: none;
}

@media screen and (min-width: 500px) {
  #selection-box {
    grid-template-columns: auto auto auto auto;
  }

  #selection-box #forth {
    display: block;
  }
}

.selection-item {
  font-size: 12px;
  display: flex;
  flex-direction: column;
}

.selection-item .cost {
  display: flex;
  flex-direction: row;
  align-items: center;
}
```

editProfile.css

> frontend/src/pages/stylesheets/editProfile.css Written by Eugene Au

```
#editProfile {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
  color: black;
}

#editProfile #content {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}

#editProfile #main {
  width: calc(100% - 40px);
  max-width: 500px;
}

#editProfile #profilePic {
  margin-top: 10px;
}

#editProfile #profilePic img {
  width: 90px;
  height: 90px;
  border: none;
  border-radius: 90px;
}

#title {
  font-size: 36px;
  font-weight: 700;
}

#editProfile form input {
  width: 100%;
  box-sizing: border-box;
  font-size: 16px;
  padding: 13px 20px;
  border: none;
  background: whitesmoke;
  border-radius: 100px;
  outline: none;
  margin-top: 5px;
}

#editProfile form .field {
  width: 100%;
  margin-top: 15px;
}
```

```
#editProfile form label {
  margin-left: 18px;
}

#editProfile form {
  width: 100%;
}

#editProfile button {
  font-size: 20px;
  font-weight: 700;
  width: 100%;
  margin-top: 30px;
  padding: 10px;
  border: none;
  border-radius: 100px;
  background: var(--primary);
}
```

feed.css

> frontend/src/pages/stylesheets/feed.css Written by Eugene Au

```
#feed {
  max-width: 500px;
  margin-left: auto;
  margin-right: auto;
  transition: filter 0.3s ease-in-out;
}

#feed.blur {
  filter: blur(5px);
}

#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 0.5s ease-out;
}

/* Animation css classes */
```

```
@keyframes polaroidFadeIn {
  from {
    margin-top: 70%;
    opacity: 0.5;
    scale: 0.7;
  }
  to {
    opacity: 1;
    margin-top: 0%;
    scale: 1;
  }
}

#feed #padding {
  padding: 80px 20px;
}

#feed .polaroid.skeleton {
  width: 100%;
  background-color: white;
  animation: none;
}

.polaroid.skeleton .padding.skeleton {
  padding: 5% 3%;
  padding-bottom: 30%;
  background-color: white;
}

.polaroid.skeleton .image.skeleton {
  width: 100%;
  aspect-ratio: 1/1;
}

/* THE LOADING EFFECT */
.skeleton {
  background-color: #e5e5e5;
  /* The shine that's going to move across the skeleton: */
  background-image: linear-gradient(
    90deg,
    rgba(255, 255, 255, 0),
    rgba(255, 255, 255, 0.5),
    rgba(255, 255, 255, 0)
  );
  background-size: 150px 100%; /* width of the shine */
  background-repeat: no-repeat; /* No need to repeat the shine effect */
  background-position: left -150px top 0; /* Place shine on the left side, with
offset on the left based on the width of the shine - see background-size */
  animation: shine 2s ease infinite; /* increase animation time to see effect in
'slow-mo' */
}

@keyframes shine {
  to {
```

```
    /* // Move shine from left to right, with offset on the right based on the
width of the shine - see background-size */
    background-position: right -150px top 0;
  }
}
```

login.css

> frontend/src/pages/login/feed.css Written by Eugene Au

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

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

/* 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 */
#login #spacer {
  padding: 70px 0;
  overflow: scroll;

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

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

/* The title of the page */
#login #title {
  font-size: 36px;
  font-weight: 700;
```

```
  color: black;
  display: flex;
  justify-content: center;
  align-items: center;

  margin-bottom: 22px;
}

/* 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%;
  margin-top: 25px;
}

/* 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";
}

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

#login #message {
  width: 100%;
}

label.error {
  color: red;
  margin-left: 5px;
}
```

map.css

> frontend/src/pages/map/feed.css Written by Adam George

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

#map-content.blur {
  filter: blur(5px);
}

/* 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, 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

```
/* center the content */
#displayProfile {
  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 */
#profile #spacer {
  padding: 70px 0;
  overflow: scroll;

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

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

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

#profile #user-icon img {
```

```
  width: 120px;
  height: 120px;
  border: none;
  border-radius: 120px;
}

#profile #user-icon {
  width: 120px;
  height: 120px;
  border-radius: 50%;
  margin: 0 auto;
  display: block;
  margin-top: 10px;
}

#socials {
  display: flex;
  justify-content: center;
  align-items: space-between;
  margin-top: 25px;
  min-width: 300px;

  /* spacing in bottom for visual purposes */
  margin-bottom: 20vh;
}

/* spacing betweeen the icons */
.social-icon:not(:last-child) {
  margin-right: 10px;
}

.social-icon {
  display: flex;
  justify-content: center;
  align-items: center;
  padding: 5px 10px;
  border: 1px solid #000;
  border-radius: 100px;
  font-size: 12px;
  text-decoration: none;
  font-weight: 700;
  color: black;
}

a {
  text-decoration: none;
}

/* spacing between the icon and text */
.social-icon img {
  margin-right: 5px;
}

#profile #title {
```

```
  text-align: center;
  font-size: 20px;
  font-weight: 700;
}

#profile #bio {
  margin-top: 30px;
  text-align: left;
  line-height: 1.3;
  max-width: 350px;
  margin-left: auto;
  margin-right: auto;
}
```

register.css

> frontend/src/pages/stylesheets/register.css Written by Eugene Au

```
/* center the content */
#displayRegister {
  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 */
#register #spacer {
  padding: 70px 0;
  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: calc(21px + 25px);
}

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

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

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

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

#register #form button {
  width: 100%;

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

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

#register #form .inline-link{
  font-weight: 500;
  display: inline-block;
}

#register #form .checkmark-label{
  display: inline;
}
```

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%);
}
```

termsConditions.css

> frontend/src/pages/stylesheets/termsConditions.css Written by Adam George

```
.terms-conditions {
    color: black;
    margin-top: 60px;
    margin-bottom: 60px;
    padding: 20px 8vw;
}

.terms-conditions h1 {
    font-size: 2rem;
    margin-bottom: 1rem;
}

.terms-conditions h2 {
    font-size: 1.2rem;
}

.terms-conditions a {
    color: black;
}

.terms-conditions .go-back {
    display: flex;
    align-items: center;
```

```
    text-decoration: none;
    font-weight: 500;
}

.terms-conditions .go-back img {
    margin-right: 2px;
}
```

features

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(redirect = true) {

    // cookies.remove('token');

    if (cookies.get('token') === undefined) {
        console.log("redirecting")
        console.log(redirect);
        if (redirect) {
            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 response;
    } catch (error) {
        if (error.response) {
            if (error.response.data.detail === "Invalid token.") {
```

```
                cookies.remove('token');
                console.log("not logged in");
                if (redirect) {
                    console.log("redirecting")
                    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);
                alert("Internal server error");
            }
        } else {
            console.log(error)
            alert("Cannot connect to the server");
        }
        return false
    }

}

export default CheckLogin;
```

dasherboardCard.jsx

> frontend/src/features/dasherboardCard.jsx Written by Adam George

```
import exitimg from '../assets/map/close.svg';
import './stylesheets/dashboardCard.css';

const DashboardCard = ({id, username, isSuperUser, image, caption, datetime,
location, userid, setPostView, deletePost, deleteUser}) => {

    function formatDate(dateString) {
        const options = { hour: '2-digit', minute: '2-digit', day: '2-digit',
month: '2-digit', year: 'numeric' };
        const date = new Date(dateString);
        const formattedDate = `${date.toLocaleTimeString('en-GB', options)}`;

        return formattedDate;
    }

    const date = formatDate(datetime);

    return (
        <div className="dashboard-card">
            <button className="exit-button" onClick={() => setPostView(false)}>
<img src={exitimg}></img></button>
            <img src={"http://127.0.0.1:8000" + image}></img>
            <p>Caption: {caption}</p>
```

```
            <p>Location: {location}</p>
            <p>Author: {username}</p>
            <p>Date posted: {date}</p>
            <div id="button-shelf">
                <button type='button' onClick={() => deletePost(id)}
classNameName="delete-button">Delete Post</button>
                <button type='button' disabled={isSuperUser} onClick={() =>
deletePost(banUser)} classNameName="delete-button">Ban User</button>
            </div>
        </div>
    );
}

export default DashboardCard;
```

DrawerDown.jsx

> frontend/src/features/DrawerDown.jsx Written by Eugene Au, 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;
```

ErrorBox.jsx

> frontend/src/features/ErrorBox.jsx Written by Eugene Au

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

function ErrorBox() {

    return (
        <div className="error-box">
            <div className="error-box__title"></div>
            <div className="error-box__message">heelo</div>
```

```
        </div>
    );
}

export default ErrorBox;
```

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 calendericon from "../assets/footer/calendar.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="/challenge">
          {/* if the user is on the top page, add backdrop to the icon */}
          <div
            className="backdrop"
            style={{
              background: location.pathname === "/challenge" ? "#00DCA5" : "none",
```

```
            }}
          >
            <img src={calendericon} id="top-icon" alt="top-icon" />
          </div>
          <div className="footer-text">challenge</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>
  );
}
```

header.jsx

> frontend/src/features/header.jsx Written by Eugene Au

```
import { useEffect, useState } from "react";
import logo from "../assets/logo-notext.png";
import usericon from "../assets/header/user-icon.jpg";
```

```
import user from "../assets/header/user.svg";
import settings from "../assets/header/setting.svg";
import "./stylesheets/header.css";
import { Link } from "react-router-dom";
import CheckLogin from "./CheckLogin";
import { useLocation } from "react-router-dom";
import Coin from '../assets/challenge/coin.png';
import axios from "axios";
import Cookies from "universal-cookie";
import Logout from "../assets/header/logout.svg";

const cookies = new Cookies();

function Header() {
  const location = useLocation();

  const [showMenu, setShowMenu] = useState(false);
  const [userIcon, setUserIcon] = useState(usericon)
  const [showIcon, setShowIcon] = useState(false)
  const [coins, setCoins] = useState(0)

  const toggleMenu = () => {
    setShowMenu(!showMenu);
  }

  useEffect(() => {
    document.addEventListener("click", (e) => {
      if (e.target.id !== "user-icon") {
        setShowMenu(false);
      }
    });


    const getIcon = async () => {
      let response = await CheckLogin(false);
      // if the user is not logged in, no icon will be shown
      if (response === false) {
        setShowIcon(false)
      }
      else {
        setShowIcon(true)
      }
      return response.data
    }

    getIcon().then((user) => {
      // if the user has not set a profile picture, the default one will be used
      if (user.profilePicture !== "NULL") {
        setUserIcon(user.profilePicture)
      }
      setCoins(user.coins)
    });
```

```
  }, [location.pathname]);

  const logout = async () => {
    let token = cookies.get('token')
    try {
      const response = await axios.get(
        "http://127.0.0.1:8000/api/logout/",
        {
          headers: {
            "Content-Type": "application/json",
            "Authorization": `Token ${cookies.get('token')}`,
          },
        }
      );

      cookies.remove('token');
      window.location.reload();
    } catch (error) {
      console.error("Error occurred:", error);
      if (error.response) {
        console.log(error.response)
        alert("internal server error, please try again later")
      } else {
        alert("cannot connect to the server")
      }
    }
  }

  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 */}
        <div>
          {showIcon &&
            (
              <div style={{ display: "flex", alignItems: "center" }}>
                <div id="coin">
                  <img src={Coin} width={"15px"} height={"15px"} />
                  <div>{coins}</div>
                </div>
                <img src={userIcon} id="user-icon" width={"25px"} height={"25px"}
style={{ border: "none", borderRadius: "25px" }} onClick={toggleMenu} />
              </div>
            )
          }

          <ul id="slide-menu" className={showMenu ? "show" : ""}>
            <Link to="/profile">
```

```
              <li id="menu" ><img src={user} width={"16px"} height={"16px"} style=
{{ marginRight: "5px" }} /><div >profile</div></li></Link>
            <Link to="/editProfile">
              <li id="menu" ><img src={settings} width={"16px"} height={"16px"}
style={{ marginRight: "5px" }} /><div >settings</div></li>
            </Link>
            <li id="logout" onClick={logout}><img src={Logout} width={"16px"}
height={"16px"} style={{ marginRight: "5px" }} /><div >logout</div></li>
          </ul></div>
      </div>
    </header >
  );
}

export default Header;
```

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";

import { useState } from "react";

function Polaroid({ src, func, rotation, caption, shadow = true }) {

  const [loadingImage, setLoadingImage] = useState(true)
  const img = new Image()
  img.src = src
  img.onload = () => {
    setLoadingImage(false)
  }
  return (
    <div
      className={shadow ? "polaroid shadow" : "polaroid"}
      style={{ transform: `rotate(${rotation}deg)` }}
      onClick={func}
    >
      <div className="padding">
        {loadingImage ? <div className="image skeleton" style={{ aspectRatio: 4 /
5 }}></div>
          :

          <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,
showBottomBar = true }) {
  return (
    <div>
      <div
        className={isActive ? "display active" : "display"}
        onClick={leaveFunction}
      >
        <div className="post-wrapper">
          <div className="post">
            <div className="spacer">
              <InteractivesTop />
              <div className="image">
                <Polaroid src={image} caption={caption} />
              </div>
              {showBottomBar && (
                <Interactives location={location} isActive={isActive} />
              )
              }
            </div>
          </div>
        </div>
      </div>
    </div>
  );
}

export default PostView;
```

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;
```

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/interactivesTop.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;
```

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;
```

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;
```

Geolocation.js

> 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;
```

stylesheets

dasherboardCard.css

> frontend/src/features/stylesheets/dasherboardCard.css Written by Adam George

```
.dashboard-card {
    display: flex;
    flex-direction: column;
    width: 300px;
    padding: 1em;
    border: 1px solid #e0e0e0;
    color: black;
}

.dashboard-card .exit-button {
    background-color: transparent;
    border: none;
    align-self: flex-end;
    margin-right:  -0.5em;
    margin-top:  -0.2em;
    cursor: pointer;
}

.dashboard-card img {
    margin-bottom: 1em;
}

.dashboard-card p {
    margin: 0;
    font-size: 1em;
    font-weight: 500;
}

.dashboard-card #button-shelf {
    margin-top: 1em;
    display: flex;
    gap: 5px;
}

.dashboard-card .delete-button {
    background: rgb(248, 70, 70);
    font-family: Outfit, sans-serif;
    font-weight: 500;
    border: none;
    padding: 0.2em 1em;
```

```
    border-radius: 50px;
}

.dashboard-card .delete-button:hover {
    background: rgb(255, 0, 0);
    cursor: pointer;
}

.dashboard-card .delete-button:disabled {
    background: rgb(238, 128, 128);
    cursor: none;
}
```

errorBox.css

> frontend/src/features/stylesheets/errorBox.css Written by Eugene Au

```
.error-box {
  display: none;
  position: fixed;
  width: 100vw;
  height: 100vh;
  background: rgba(12, 12, 12, 0.7);
}
```

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;
}

#slide-menu {
  display: none;
  position: fixed;
  right: 5px;
  background: white;
  color: black;
  list-style: none;
  padding: 10px;
  border: none;
  border-radius: 12px;
}

header #slide-menu.show {
  display: block;
  animation: slide 0.3s ease-in-out;
}

@keyframes slide {
  0% {
    transform: translateY(10px);
    opacity: 0;
  }
  100% {
    transform: translateY(0);
    opacity: 1;
  }
}

#slide-menu #menu {
  display: flex;
  align-items: center;
```

```
}

header #coin {
  display: flex;
  align-items: center;
  font-size: 12px;
  line-height: 1;
  background-color: rgb(0, 220, 165);
  border: none;
  border-radius: 12px;
  padding: 5px 7px;
  margin-right: 10px;
}
header #coin :first-child {
  margin-right: 2px;
}
#logout {
  display: flex;
  align-items: center;
}
```

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;

  z-index: 10;
}

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;
}
```

InitMap.css

> frontend/src/features/stylesheets/InitMap.css Written by Eugene Au

```
.splash {
  position: absolute;
  height: 100vh;
  width: 100vw;
  background-color: white;
  z-index: 9;
  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;
} */
```

polroid.css

> frontend/src/features/stylesheets/polroid.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;
}

.polaroid.shadow {
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

  /* space for the width to not get the shadow to cut off */
  width: calc(100% - 8px);
  margin-left: auto;
  margin-right: auto;
}
```

```
/* 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;
}

/* THE LOADING EFFECT */
.skeleton {
  background-color: #e5e5e5;
  /* The shine that's going to move across the skeleton: */
  background-image: linear-gradient(
    90deg,
    rgba(255, 255, 255, 0),
    rgba(255, 255, 255, 0.5),
    rgba(255, 255, 255, 0)
  );
  background-size: 150px 100%; /* width of the shine */
  background-repeat: no-repeat; /* No need to repeat the shine effect */
  background-position: left -150px top 0; /* Place shine on the left side, with
offset on the left based on the width of the shine - see background-size */
  animation: shine 2s ease infinite; /* increase animation time to see effect in
'slow-mo' */
}

@keyframes shine {
  to {
    /* // Move shine from left to right, with offset on the right based on the
width of the shine - see background-size */
    background-position: right -150px top 0;
  }
}
```

postView.css

> frontend/src/features/stylesheets/polroid.css 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,
showBottomBar = true }) {
  return (
    <div>
      <div
        className={isActive ? "display active" : "display"}
        onClick={leaveFunction}
      >
        <div className="post-wrapper">
          <div className="post">
            <div className="spacer">
              <InteractivesTop />
              <div className="image">
                <Polaroid src={image} caption={caption} />
              </div>
              {showBottomBar && (
                <Interactives location={location} isActive={isActive} />
              )
              }
            </div>
          </div>
        </div>
      </div>
    </div>
  );
}

export default PostView;
```

End