2022년 1학기 오픈소스SW및실습 Term Project

조 이름: 만수무강

가) 프로그램의 목적 및 개요

최근 환경문제로 인해 세계 곳곳에서 피해를 입는 사례가 증가하고 있다. [1]2022년 5월 자 미국에서는 동부지역에서 35도 폭염인 동시에 서부에는 50cm 눈이 내리는 등등 세계 곳곳에서 환경문제가 발견되는 빈도수가 점차 늘어나고 있는 추세이다. 한국에서도 마찬가지로 미세먼지, 과도한 폭염 등등 우리 실생활에 환경문제가 영향을 끼치기 시작하였다. 이에 따라 사람들은 개개인들이 환경문제에 관심을 가지고 사소한 일이라도 환경에 도움이 될 만한 활동들을 실천해야 한다는 목소리가 높아지고 있다.

이러한 인식을 반영하여 만든 원칙이 바로 '제로웨이스트(Zero Waste)' 이며, [2]'제로 웨이스트'란 모든 제품, 포장 및 자재를 태우지 않고, 환경이나 인간의 건강을 위협할 수 있는 토지, 해양, 공기로 배출하지 않으며 책임 있는 생산, 소비, 재사용 및 회수를 통해 모든 자원을 보존하도록 돕는 원칙이다.

만수무강조는 '제로 웨이스트' 원칙에 의거한 '5R 운동'을 사람들이 보다 쉽게 접근하고, 환경에 관심을 가질 수 있도록 돕기 위해 '#그린태그'라는 환경 어플을 만들기로 결정하였다. '#그린태그' 어플에서 다이어리파트에는 환경과 관련된 본인의 생활에 대하여 기록할 수 있도록 만들고, 액티비티 파트에서는 제로웨이스트 기준 5R(Refuse, Reuse, Reduce, Recycle, Rot)을 이해하고, 자신이 행한 5R 관련 활동을 점수화 하여 차트를 이용하여 실

---

[1] http://www.munhwa.com/news/view.html?no=20220523MW075645419840

[2]https://ko.wikipedia.org/wiki/%EC%A0%9C%EB%A1%9C_%EC%9B%A8%EC%9D%B4%EC%8A%A4%ED%8A%B8

천률을 확인하게 만들며, 환경 관련 뉴스들을 통해 경각심을 갖을 수 있도록 도와줄 수 있다.

'#그린태그' 어플리케이션은 사람들이 보다 환경문제에 쉽게 접할 수 있도록 도와주고 개개인의 제로웨이스트에 대한 실천을 증진시키는 목적으로 만들어졌다.
서론 : 환경 문제 -> 해결 방법 -> 개인 실천이 가능한 제로웨이스트 -> 앱으로 개발
본론 : 다이어리로 본인 생활 기록, 액티비티에서 제로웨이스트 기준 5R을 이해하고, 차트로 점수화 , 실천률 확인, 관련 뉴스를 통해 경각심 갖기
결론 : 이 앱은 환경문제에 쉽게 접근하게 도와주고 제로웨이스트에 대한 실천을 도와줌

나) 각 팀원의 역할

- 방희연 : navigation, 정보페이지 뉴스 기능 구현, 전체 컴포넌트 오류 수정, 전체 페이지, 그린태그 기획, 디자인, github 관리, 프로젝트 총괄
- 김정헌 : Check페이지 5R 점수 계산 컴포넌트 기능 구현, 개인 점수 가중치 체계 설계, 프로젝트 일정 관리, 그린태그 기획 및 네이밍
- 고태규 : Check페이지 5R이란 컴포넌트 구현, 이미지 및 자료 수집 및 정리, 그린태그 기획
- 백지연 : 정보페이지 Chart 구현, 디자인, 그린태그 기획
- 석상범 : Diary 페이지 navigation flow 기획, Diary 페이지 전체 컴포넌트 기능 구현
- 배인범 : Diary 페이지 기능 구현, 오류 수정,

다) 프로그램의 구조 및 설명

a. 구조

버튼 탭으로 접근, 큰 구조는 1)Diary, 2)Check, 3)News 세 가지로 이루어져 있음.

1) DIary는 일기를 적는 탭과 달력을 통해 일기를 확인하는 탭, 검색어로 일기를 찾는 탭으로 구성
2) Check는 나의 활동을 점수화 할 수 있는 페이지로 이동하는 버튼, 5R에 대한 정보를 알 수 있는 버튼으로 구성
3) News는 점수들을 차트로 확인할 수 있는 영역과 제로웨이스트 관련 뉴스들을 확인할 수 있는 영역으로 구성

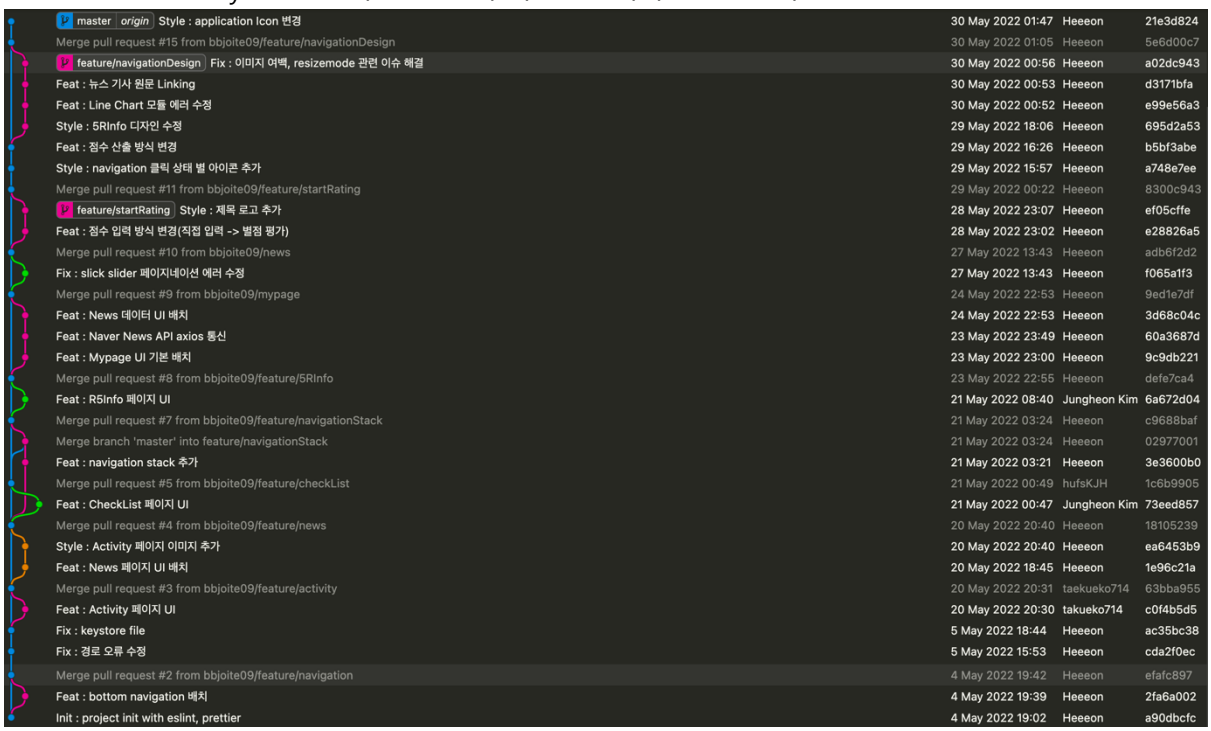
b. Github

Github는 git flow 방식에서 착안하여 master 브랜치로 부터 기능 단위 브랜치를 파생시켜 진행하였다. 각 기능 구현이 완료된 이후에는 원격 브랜치에 push 하여 pull Request를 생성하였고, 이를 상호 확인하여 master branch 에 merge 하는 방식으로 충돌을 최소화하였다.

커밋 메시지는 팀원 전체가 다음과 같은 형식으로 통일하여 작성함으로써, 진행사

항을 쉽게 확인할 수 있고, 문제 발생 시 revert 가 용이하도록했다.

- Feat : 기능 추가 내용
- Fix : 에러 수정 내용
- Style : 코드에 영향을 주지 않는 디자인 변경 내용으로



라) 소스 코드 전체

0. Navigation Stack

Bottom Tab 네비게이션

src/elements/AllStack.js

```
import React from 'react';
import {createBottomTabNavigator} from '@react-navigation/bottom-tabs';
import {createNativeStackNavigator} from '@react-navigation/native-stack';
import Information from '../components/Information';
import Activity from '../components/Activity';
import CheckList from './activity/CheckList';
import R5Info from './activity/R5Info';
import {Image, Text} from 'react-native';
import Icon from 'react-native-vector-icons/MaterialIcons';

import SearchHeader from '../components/SearchHeader';
import FeedsScreen from '../components/FeedsScreen';
import CalendarScreen from '../components/CalendarScreen';
import SearchScreen from '../components/SearchScreen';
import WriteScreen from '../components/WriteScreen';
const Tab = createBottomTabNavigator();
```

```
//하단의 탭을 구성하는 것들을 선언한 함수
const BottomTabs = () => {
  return (
    <Tab.Navigator
      screenOptions={{
        tabBarShowLabel: false,
      }}>
      <Tab.Screen
        name="녹색일기 쓰기"
        component={FeedsScreen}
        options={{
          tabBarIcon: ({focused}) => (
            <Icon name="view-stream" size={24} color={'gray'} />
          ),
        }}
      />
      <Tab.Screen
        name="달력"
        component={CalendarScreen}
        options={{
          tabBarIcon: ({focused}) => (
            <Icon name="event" size={24} color={'gray'} />
          ),
        }}
      />
      <Tab.Screen
        name="Search"
        component={SearchScreen}
        options={{
          title: '검색',
          tabBarIcon: ({focused}) => (
            <Icon name="search" size={24} color={'gray'} />
          ),
          headerTitle: () => <SearchHeader />,
        }}
      />
      <Tab.Screen
        name="녹색활동"
        component={Activity}
        options={{
          tabBarIcon: ({focused}) =>
            !focused ? (
              <Image
                source={require('../assets/shared/activity_no.png')}
                style={{width: 30, height: 30}}
              />
            ) : (
              <Image
                source={require('../assets/shared/activity.png')}
                style={{width: 30, height: 30}}
              />
            ),
        }}
      />
      <Tab.Screen
        name="정보"
```

```
        options={{
          tabBarIcon: ({focused}) =>
            !focused ? (
              <Image
                source={require('../assets/shared/info_no.png')}
                style={{width: 30, height: 30}}
              />
            ) : (
              <Image
                source={require('../assets/shared/info.png')}
                style={{width: 30, height: 30}}
              />
            ),
        }}
      />
    </Tab.Navigator>
  );
};

const Stack = createNativeStackNavigator();

const AllStack = () => {
  return (
    <Stack.Navigator initialRouteName="BottomTabs">
      <Stack.Screen
        name="BottomTabs"
        component={BottomTabs}
        options={{headerShown: false}}
      />
      <Stack.Screen
        name="Activity"
        component={Activity}
        options={{headerShown: false}}
      />
      <Stack.Screen
        name="CheckList"
        component={CheckList}
        options={{
          headerShown: true,
          headerTitle: '',
        }}
      />
      <Stack.Screen
        name="R5Info"
        component={R5Info}
        options={{
          headerShown: true,
          headerTitle: '',
        }}
      />
      <Stack.Screen
        name="WriteScreen"
        component={WriteScreen}
        options={{
          headerShown: false,
        }}
      />
```

```
  </Stack.Navigator>
   );
 };
 export default AllStack;
```

App.js

```
import React from 'react';
import {DefaultTheme, NavigationContainer} from '@react-navigation/native';
import AllStack from './src/elements/AllStack';
import {SearchContextProvider} from './src/contexts/SearchContext';
import {LogContextProvider} from './src/contexts/LogContext';
const App = () => {
  return (
    // navigation 배경을 흰색으로 설정해준다.
    <NavigationContainer
      theme={{
        ...DefaultTheme,
        colors: {
          ...DefaultTheme.colors,
          background: 'white',
        },
      }}>
      <SearchContextProvider>
        <LogContextProvider>
          {/* 하단 탭으로 배치한 전체 UI 를 넣어준다 */}
          <AllStack />
        </LogContextProvider>
      </SearchContextProvider>
    </NavigationContainer>
  );
};

export default App;
```

1. Diary 파트 코드

Diary 파트의 Calendar탭 코드

src/assests/components/ CalendarScreen.js

```
import React, {useMemo, useState} from 'react';
import {StyleSheet} from 'react-native';
//date-fns를 사용하여 시간을 표현가능
import {format} from 'date-fns';
import CalendarView from '../components/CalendarView';
```

```
import {useLog} from '../contexts/LogContext';
import FeedList from '../components/FeedList';

function CalendarScreen() {
  const {logs} = useLog();
  const [selectedDate, setSelectedDate] = useState(
    format(new Date(), 'yyyy-MM-dd'),
  );
// 시간을 표시해주는 함수 선언
  const markedDates = useMemo(
    () =>
      logs.reduce((acc, current) => {
        const formattedDate = format(new Date(current.date), 'yyyy-MM-dd');
        acc[formattedDate] = {marked: true};
        return acc;
      }, {}),
    [logs],
  );
 // 선택한 시간을 저장해주는 함수
  const filteredLogs = logs.filter(
    log => format(new Date(log.date), 'yyyy-MM-dd') === selectedDate,
  );
 //캘린더 화면을 표시
  return (
    <FeedList
      logs={filteredLogs}
      ListHeaderComponent={
        <CalendarView
          markedDates={markedDates}
          selectedDate={selectedDate}
          onSelectDate={setSelectedDate}
        />
      }
    />
  );
}
// 캘린더 탭에서 사용할 스타일 선언
const styles = StyleSheet.create({
  block: {},
  text: {
    padding: 16,
    fontSize: 24,
  },
  rectangle: {width: 100, height: 100, backgroundColor: 'black'},
});
 //캘린더
export default CalendarScreen;
```

src/assests/components/ CalendarView.js

```
import React from 'react';
//캘린더를 사용할 수있게해주는 react-native-calendars import
```

```
import {Calendar} from 'react-native-calendars';
import {StyleSheet} from 'react-native';
// 선택한 날짜를 저장해주는 함수
function CalendarView({markedDates, selectedDate, onSelectDate}) {
  const markedSelectedDate = {
    ...markedDates,
    [selectedDate]: {
      selected: true,
      marked: markedDates[selectedDate]?.marked,
    },
  };
  return (
    <Calendar
      style={styles.calendar}
      markedDates={markedSelectedDate}
      onDayPress={day => {
        onSelectDate(day.dateString);
      }}
//캘린더의 스타일을 선언
      theme={{
        selectedDayBackgroundColor: '#009688',
        arrowColor: '#009688',
        dotColor: '#009688',
        todayTextColor: '#009688',
      }}
    />
  );
}

const styles = StyleSheet.create({
  calendar: {
    borderBottomWidth: 1,
    borderBottomColor: '#e0e0e0',
  },
});

export default CalendarView;
```

Diary 파트의 Write탭 코드

src/assests/components/ WriteScreen.js

```
import React, {useState} from 'react';
//safeareaview를 통해 여백을 만들어 줌
import {SafeAreaView} from 'react-native-safe-area-context';
import {useNavigation, useRoute} from '@react-navigation/native';
//keyboardavoidingview를 통해 키보드로 타이핑하는 화면을 가리지 않게해줌
import {StyleSheet, KeyboardAvoidingView, Platform, Alert} from 'react-native';

import {useLog} from '../contexts/LogContext';
import WriteHeader from '../components/WriteHeader';
import WriteEditor from '../components/WriteEditor';
```

```
//화면을 구성하는 요소를 구성해 WriteScreen 함수 선언
function WriteScreen() {
  const {params} = useRoute();

  const log = params?.log;

  console.log(log);

  const [title, setTitle] = useState(log?.title ?? '');
  const [body, setBody] = useState(log?.body ?? '');
  const {onCreate, onModify, onRemove} = useLog();
  const [date, setDate] = useState(log ? new Date(log.date) : new Date());

  const navigation = useNavigation();

  const onSave = () => {
    if (log) {
      onModify({
        id: log.id,
        date: date.toISOString(),
        title,
        body,
      });
    } else {
      onCreate({title, body, date: date.toISOString()});
    }
    navigation.pop();
  };

  const onAskRemove = () => {
    if (!log?.id) {
      return;
    }

    Alert.alert(
      '삭제',
      '정말로 삭제하시겠어요?',
      [
        {text: '취소', style: 'cancel'},
        {
          text: '삭제',
          style: 'destructive',
          onPress: () => {
            onRemove(log?.id);
            navigation.pop();
          },
        },
      ],
      {cancelable: true},
    );
  };
   return (
    <SafeAreaView style={styles.block}>
      <KeyboardAvoidingView
```

```
        behavior={Platform.OS === 'ios' ? 'padding' : undefined}>
        <WriteHeader
          onSave={onSave}
          onAskRemove={onAskRemove}
          isEditing={!!log}
          date={date}
          onChangeDate={setDate}
        />
        <WriteEditor
          title={title}
          body={body}
          onChangeTitle={setTitle}
          onChangeBody={setBody}
        />
      </KeyboardAvoidingView>
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({
  block: {
    flex: 1,
    backgroundColor: 'white',
  },
  avoidingView: {
    flex: 1,
  },
});
export default WriteScreen;
```

src/assests/components/ WriteHeader.js

```
import React, {useState, useReducer} from 'react';
import {useNavigation} from '@react-navigation/native';
import {View, StyleSheet, Pressable, Text} from 'react-native';
import {format} from 'date-fns';
import {ko} from 'date-fns/locale';
//datetimepickermodal을 통해 날짜를 선택할 수있게함
import DateTimePickerModal from 'react-native-modal-datetime-picker';
// TransparentCircleButton을 통해 버튼을 추가해 데이터 추가 할때 사용함
import TransparentCircleButton from './TransparentCircleButton';

const initialState = {mode: 'date', visible: false};
// 실행이 되면보이고 아니면 보이지 않는 reducer함수 생성
function reducer(state, action) {
  switch (action.type) {
    case 'open':
      return {
        mode: action.mode,
        visible: true,
      };
    case 'close':
      return {
```

```
        ...state,
        visible: false,
      };
    default:
      throw new Error('Unhandled action type');
  }
}
//정한 날짜에 기록하기 전 화면에서 전환 할수있게 하는 writeheader함수
function WriteHeader({onSave, onAskRemove, isEditing, date, onChangeDate}) {
  const [state, dispatch] = useReducer(reducer, initialState);
  const open = mode => dispatch({type: 'open', mode});
  const close = () => dispatch({type: 'close'});
  const naviagtion = useNavigation();

  const onGoBack = () => {
    naviagtion.pop();
  };

  const onConfirm = selectedDate => {
    close();
    onChangeDate(selectedDate);
  };

  return (
    <View style={styles.block}>
      <TransparentCircleButton
        onPress={onGoBack}
        name="arrow-back"
        color="#424242"
      />

      <View style={styles.buttons}>
        {isEditing && (
          <TransparentCircleButton
            name="delete-forever"
            color="#ef5350"
            hasMarginRight
            onPress={onAskRemove}
          />
        )}
        <TransparentCircleButton
          name="check"
          color="#009688"
          onPress={onSave}
        />
      </View>
      <View style={styles.center}>
        <Pressable onPress={() => open('date')}>
          <Text>{format(new Date(date), 'PPP', {locale: ko})}</Text>
        </Pressable>
        <View style={styles.separator} />
        <Pressable onPress={() => open('time')}>
          <Text>{format(new Date(date), 'p', {locale: ko})}</Text>
        </Pressable>
      </View>
      <DateTimePickerModal
```

```
        isVisible={state.visible}
        mode={state.mode}
        onConfirm={onConfirm}
        onCancel={close}
        date={date}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  block: {
    height: 48,
    paddingHorizontal: 8,
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'space-between',
  },

  buttons: {
    flexDirection: 'row',
    alignItems: 'center',
  },
  center: {
    position: 'absolute',
    left: 0,
    right: 0,
    top: 0,
    bottom: 0,
    alignItems: 'center',
    justifyContent: 'center',
    zIndex: -1,
    flexDirection: 'row',
  },
  separator: {
    width: 8,
  },
});

export default WriteHeader;
```

src/assests/components/ WriteEditor.js

```
import React, {RefObject, useRef} from 'react';
import {View, StyleSheet, TextInput} from 'react-native';
//글을 쓸 수있게 해주는 함수인 writeeditor 함수 생성
function WriteEditor({title, body, onChangeTitle, onChangeBody}) {
  const bodyRef = useRef(null);
// 기록 폼을 보여줌
  return (
    <View style={styles.block}>
      <TextInput
        placeholder="제목을 입력하세요"
        onChangeText={onChangeTitle}
```

```
        value={title}
        returnKeyType="next"
        style={styles.titleInput}
        onSubmitEditing={() => {
          bodyRef.current?.focus();
        }}
      />
      <TextInput
        placeholder="오늘 실천한 제로 웨이스트 활동을 기록해주세요."
        style={styles.bodyInput}
        multiline
        onChangeText={onChangeBody}
        value={body}
        textAlignVertical="top"
        ref={bodyRef}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  block: {
    flex: 1,
    padding: 16,
  },
  titleInput: {
    paddingVertical: 0,
    fontSize: 18,
    marginBottom: 16,
    color: '#263238',
    fontWeight: 'bold',
  },
  bodyInput: {
    flex: 1,
    fontSize: 16,
    paddingVertical: 0,
    color: '#263238',
  },
});

export default WriteEditor;
```

src/components/FeedList.js

```
import React from 'react';
//flatlist를 통해 데이터가 화면을 벗어났을 때 scroll 사용가능
//nativescrollevent,nativesyntheticevent를 통해 사용자 지정 애니메이션 적용을 위한 임포트
import {
  FlatList,
  NativeScrollEvent,
  NativeSyntheticEvent,
  StyleSheet,
```

```
  View,
} from 'react-native';

// import {Log} from '../contexts/LogContext';
import FeedListItem from './FeedListItem';
// 데이터 양이 많아지면 스크롤을 생성하고, 그 동안 기록을 보여주는 feedlist함수
function FeedList({logs, onScrolledToBottom, ListHeaderComponent}) {
  const onScroll = e => {
    if (!onScrolledToBottom) {
      return;
    }

    const {contentSize, layoutMeasurement, contentOffset} = e.nativeEvent;

    const distanceFromBottom =
      contentSize.height - layoutMeasurement.height - contentOffset.y;

    if (
      distanceFromBottom < 72 &&
      contentSize.height > layoutMeasurement.height
    ) {
      onScrolledToBottom(true);
    } else {
      onScrolledToBottom(false);
    }
  };

  return (
    <FlatList
      data={logs}
      style={styles.block}
      renderItem={({item}) => <FeedListItem log={item} />}
      keyExtractor={log => log.id}
      ItemSeparatorComponent={() => <View style={styles.separator} />}
      onScroll={onScroll}
      ListHeaderComponent={ListHeaderComponent}
    />
  );
}

const styles = StyleSheet.create({
  block: {flex: 1},
  separator: {
    backgroundColor: '#e0e0e0',
    height: 1,
    widht: '100%',
  },
});

export default FeedList;
```

Diary 파트의 Search탭 코드

src/assests/components/SearchScreen.js

```
import React from 'react';

import {StyleSheet, View, Text} from 'react-native';
import EmptySearchResult from '../components/EmptySearchResult';
import FeedList from '../components/FeedList';
import {useLog} from '../contexts/LogContext';

import {useSearch} from '../contexts/SearchContext';
// 기록들을 검색하는 기능을 수행 함수 searchscreen
function SearchScreen() {
  const {keyword} = useSearch();
  const {logs} = useLog();

  const filtered =
    keyword === ''
      ? []
      : logs.filter(log =>
          [log.title, log.body].some(text => text.includes(keyword)),
        );

  if (keyword === '') {
    return <EmptySearchResult type="EMPTY_KEYFOUND" />;
  }

  if (filtered.length === 0) {
    return <EmptySearchResult type="NOT_FOUND" />;
  }

  return (
    <View style={styles.block}>
      <FeedList logs={filtered} />
    </View>
  );
}
const styles = StyleSheet.create({
  block: {
    flex: 1,
  },
});
export default SearchScreen;
```

src/components/FeedsScreen.js

```
import React, {useState} from 'react';
import {StyleSheet, View, TextInput, Text} from 'react-native';
import FeedList from './FeedList';
```

```
// floatingwritebutton을 이용해 기록을 추가하려고 할때의 버튼으로 사용함
import FloatingWriteButton from './FloatingWriteButton';
import {useLog} from '../contexts/LogContext';
// 기록을 추가하기 전의 화면을 보여주는 함수 feedscreen
function FeedsScreen() {
  const {logs} = useLog();

  console.log(logs);

  const [hidden, setHidden] = useState(false);

  const onScrolledToBottom = isBottom => {
    if (hidden !== isBottom) {
      setHidden(isBottom);
    }
  };

  return (
    <View style={styles.block}>
      <FeedList logs={logs} onScrolledToBottom={onScrolledToBottom} />
      <FloatingWriteButton hidden={hidden} />
    </View>
  );
}

const styles = StyleSheet.create({
  block: {
    flex: 1,
  },
  input: {
    padding: 16,
    backgroundColor: 'white',
  },
});

export default FeedsScreen;
```

src/components/ FeedListItem.js

```
import React from 'react';
import {useNavigation} from '@react-navigation/native';
import {Platform, Pressable, StyleSheet, Text} from 'react-native';
import {format, formatDistanceToNow} from 'date-fns';
import {ko} from 'date-fns/locale';

function truncate(text) {
  const replaced = text.replace('/\n/g', ' ');

  if (replaced.length <= 100) {
    return replaced;
  }
```

```
  return replaced.slice(0, 100).concat('...');
}
//기록을 언제했는지 표기되게 하는 함수 formatdate
function formatDate(date) {
  const d = new Date(date);
  const now = Date.now();
  const diff = (now - d.getTime()) / 1000;

  if (diff < 60 * 1) {
    return '방금전';
  }
  if (diff < 60 * 60 * 24 * 3) {
    return formatDistanceToNow(d, {addSuffix: true, locale: ko});
  }

  return format(d, 'PPP EEE p', {locale: ko});
}
// 기록한 것들을 log에 저장하는 함수 feedlistitem
function FeedListItem({log}) {
  const {title, body, date} = log;

  const navigation = useNavigation();

  const onPress = () => {
    // navigation.navigate('WriteScreen', {
    //   log,
    // });
    console.log('OnPress');
  };

  return (
    <Pressable
      onPress={onPress}
      style={({pressed}) => [
        styles.block,
        Platform.OS === 'ios' && pressed && {backgroundColor: '#efefef'},
      ]}
      android_ripple={{color: '#ededed'}}>
      <Text style={styles.date}>{formatDate(date)}</Text>
      <Text style={styles.title}>{title}</Text>
      <Text style={styles.body}>{truncate(body)}</Text>
    </Pressable>
  );
}

const styles = StyleSheet.create({
  block: {
    backgroundColor: 'white',
    paddingHorizontal: 16,
    paddingVertical: 24,
  },
  date: {
    fontSize: 12,
```

```
    color: '#546e7a',
    marginBottom: 24,
  },
  title: {
    color: '#37474f',
    fontSize: 18,
    fontWeight: 'bold',
    marginBottom: 8,
  },
  body: {
    color: '#37474f',
    fontSize: 16,
    lineHeight: 21,
  },
});

export default FeedListItem;
```

src/assests/components/ SearchHeader.js

```
import React from 'react';
import {
  Pressable,
  StyleSheet,
  Text,
  TextInput,
  useWindowDimensions,
  View,
} from 'react-native';
import Icon from 'react-native-vector-icons/MaterialIcons';
import {useSearch} from '../contexts/SearchContext';
// 검색의 탭의 폼을 보여주는 searcheader
function SearchHeader() {
  const {width} = useWindowDimensions();
  const {keyword, onChangeText} = useSearch();


  return (
    <View style={[styles.block, {width: width - 32}]}>
      <TextInput
        placeholder="검색어를 입력하세요"
        value={keyword}
        onChangeText={onChangeText}
        autoFocus
        style={styles.input}
      />
      <Pressable
        style={({pressed}) => [styles.button, pressed && {opacity: 0.5}]}
        onPress={() => onChangeText('')}>
        <Icon name="cancel" size={20} color="#9e9e9e" />
      </Pressable>
    </View>
  );
}
```

```
const styles = StyleSheet.create({
  block: {
    color: 'black',
    backgroundColor: 'white',
    flexDirection: 'row',
    alignItems: 'center',
  },
  input: {
    flex: 1,
  },
  button: {
    marginLeft: 8,
  },
});

export default SearchHeader;
```

Diary 파트의 Storage, Context, FloatingWritebutton 코드

src/storages/logsStorage.js.

```
import AsyncStorage from '@react-native-community/async-storage';

const key = 'logs';
// 기록이 저장되는 장소로, 저장,로드를 했는지 알려주는 함수
const logsStorage = {
  async get() {
    try {
      const raw = await AsyncStorage.getItem(key);
      const parsed = JSON.parse(raw);
      return parsed;
    } catch (error) {
      throw new Error('Failed to load logs');
    }
  },
  async set(data) {
    try {
      await AsyncStorage.setItem(key, JSON.stringify(data));
    } catch (error) {
      throw new Error('Failed to save logs');
    }
  },
};

export default logsStorage;
```

src/contexts/LogContext.js

```
import React, {
  useState,
  useContext,
  createContext,
  useRef,
  useEffect,
} from 'react';
import {customAlphabet} from 'nanoid/non-secure';

import logsStorage from '../storages/logsStorage';

const LogContext = createContext(null);
// 기록되었던 것을 수정,제거할 수있게하는 함수 logcontextprovider
export function LogContextProvider({children}) {
  const initialLogsRef = useRef(null);
  const [logs, setLogs] = useState([]);
  const nanoid = customAlphabet('abcdefghijklmnopqrstuvwxyz0123456789', 10);

  useEffect(() => {
    (async () => {
      const savedLogs = await logsStorage.get();
      if (savedLogs) {
        initialLogsRef.current = savedLogs;
        setLogs(savedLogs);
      }
    })();
  }, []);

  useEffect(() => {
    if (logs === initialLogsRef.current) {
      return;
    }
    logsStorage.set(logs);
  }, [logs]);

  const onCreate = ({title, body, date}) => {
    const log = {
      id: nanoid,
      title,
      body,
      date,
    };
    setLogs([log, ...logs]);
  };

  const onModify = modified => {
    const nextLogs = logs.map(log => (log.id === modified.id ? modified : log));
    setLogs(nextLogs);
  };

  const onRemove = id => {
```

```
    const nextLogs = logs.filter(log => log.id !== id);
    setLogs(nextLogs);
  };

  return (
    <LogContext.Provider value={{logs, onCreate, onModify, onRemove}}>
      {children}
    </LogContext.Provider>
  );
}
//log가 있을경우 log를 없을 경우 안된다는 메시지가 나오는 함수 uselog함수
export function useLog() {
  const log = useContext(LogContext);

  if (!log) {
    throw new Error('LogContextProvider is not used');
  }

  return log;
}

export default LogContext;
```

src/contexts/SearchContext.js

```
import React, {useState, createContext, useContext} from 'react';

const SearchContext = createContext(null);
//로그에 있는 것과 타이핑한 것을 비교해주는 함수 SearchContextProvider
export function SearchContextProvider({children}) {
  const [keyword, onChangeText] = useState('');

  return (
    <SearchContext.Provider value={{keyword, onChangeText}}>
      {children}
    </SearchContext.Provider>
  );
}
//검색을 해서 있으면 그대로 없다면 메시지를 출력해주는 함수 usesearch
export function useSearch() {
  const context = useContext(SearchContext);
  if (context === null) {
    throw new Error('useSearch must be used within a SearchContextProvider');
  }
  return context;
}

export default SearchContext;
```

src/components/FloatingWriteButton.js

```
import {useNavigation} from '@react-navigation/native';
import React, {useEffect, useRef} from 'react';
import {
  Animated,
  Platform,
  Pressable,
  StyleSheet,
  View,
  Text,
} from 'react-native';
import Icon from 'react-native-vector-icons/MaterialIcons';
//기록 추가 버튼 기능을 하는 함수 FloatingWriteButton
function FloatingWriteButton({hidden}) {
  const navigation = useNavigation();

  const onPress = () => {
    navigation.navigate('WriteScreen');
  };

  const animation = useRef(new Animated.Value(0)).current;

  useEffect(() => {
    Animated.spring(animation, {
      toValue: hidden ? 1 : 0,
      useNativeDriver: true,
      tension: 45,
      friction: 5,
    }).start();
  }, [hidden, animation]);

  return (
    <Animated.View
      style={[
        styles.wrapper,
        {
          transform: [
            {
              translateY: animation.interpolate({
                inputRange: [0, 1],
                outputRange: [0, 88],
              }),
            },
          ],
          opacity: animation.interpolate({
            inputRange: [0, 1],
            outputRange: [1, 0],
          }),
        },
      ]}>
      <Pressable
        style={({pressed}) => [
          styles.button,
          Platform.OS === 'ios' && {
            opacity: pressed ? 0.6 : 1,
          },
```

```
        ]}
        android_ripple={{color: 'white'}}
        onPress={onPress}>
        <Icon name="add" size={24} style={styles.icon} />
      </Pressable>
    </Animated.View>
  );
}

const styles = StyleSheet.create({
  wrapper: {
    position: 'absolute',
    bottom: 16,
    right: 16,
    width: 56,
    height: 56,
    borderRadius: 28,
    shadowColor: '#4d4d4d',
    shadowOffset: {width: 0, height: 4},
    shadowOpacity: 0.3,
  },
  button: {
    width: 56,
    height: 56,
    borderRadius: 28,
    backgroundColor: '#009688',
    justifyContent: 'center',
    alignItems: 'center',
  },
  icon: {
    color: 'white',
  },
});

export default FloatingWriteButton;
```

2. Check 파트 코드

Check파트의 첫 페이지 코드

src/assests/components/activity.js

```
mport React from 'react';
import {
  StyleSheet,
  Text,
  View,
  Image,
  Pressable,
  Dimensions,
} from 'react-native';
import {SafeAreaView} from 'react-native-safe-area-context';
//  이미지 resource import
import r5 from '../assets/activity/r5.png';
import mainLogo from '../assets/activity/logo.png';
```

```
//Activity 창에선 R5이미지를 보여주며 5R 점수계산 페이지와 5R Information 페이지를 선
택할 수 있게 돕는다.
const Activity = ({navigation}) => {
  //navigation을 props로 받아온다.
  return (
    //하단 View는 Style에 정의 되어있는 container의 양식을 따름
    <SafeAreaView style={styles.container}>
      <Image
        //{/*main Logo Image를 불러옴. 이미지의 Style은 하단 Style 양식을 따름 */}
        source={mainLogo}
        style={{
          width: '100%',
          resizeMode: 'contain',
          alignItems: 'flex-start',
          marginTop: '-10%',
        }}
      />
      <Image
        //{/*r5 Image를 불러옴 */}
        style={{width: '90%', height: '50%', marginTop: '-10%'}}
        source={r5}
      />
      <View
        style={styles.footer}
        /* navigation bar를 감싸는 View로 style은 footer의 양식을 따름 /* CheckList
로
      이동 시키는 navigation bar로, roundButton의 양식을 따름. */
      >
        <Pressable
          style={styles.roundButton}
          onPress={() => {
            //Press시 화면 전환
            navigation.navigate('CheckList');
          }}>
          <Text style={styles.buttonText}>5R 점수 계산</Text>
        </Pressable>
        <Pressable
          /*R5Info로 이동 시키는 navigation bar로, CheckList bar와 마찬가지의 양식을
따름*/
          style={styles.roundButton}
          onPress={() => {
            //Press시 화면 전환
            navigation.navigate('R5Info');

            /*5R이란 텍스트를 보여준다. style은 buttionText 양식을 따름 */
          }}>
          <Text style={styles.buttonText}>5R 이란?</Text>
        </Pressable>
      </View>
    </SafeAreaView>
  );
};
//해당 페이지에서 사용하는 style들 선언
const styles = StyleSheet.create({
```

```
  container: {
    display: 'flex',
    width: Dimensions.get('window').width,
    height: '100%',
    alignItems: 'center',
    justifyContent: 'flex-start',
    paddingTop: 10,
    position: 'relative',
  },
  footer: {
    width: '100%',
    height: '20%',
    display: 'flex',
    alignItems: 'center',
    justifyContent: 'space-evenly',
  },
  roundButton: {
    width: '65%',
    height: '40%',
    backgroundColor: '#405C2F',
    borderRadius: 100,
    display: 'flex',
    justifyContent: 'center',
    alignItems: 'center',
    marginTop: '5%',
  },
  buttonText: {
    fontSize: 18,
    fontWeight: '600',
    color: 'white',
  },
});

export default Activity;
```

Check파트의 점수계산 탭 코드

src/elements/activity/Checklist.js

```
/* eslint-disable no-undef */
import React, {useState} from 'react';
import {StyleSheet, Text, View, Image} from 'react-native';
import {Rating} from 'react-native-ratings';
import {SafeAreaView} from 'react-native-safe-area-context';
// Image Resource import

/*CheckList 창에서는 5R 관련한 5가지 활동에 대한 사용자의 활동에 따른 평가점수를 받아
총 점수로 환산해주어 5R 활동량을 가시화
하도록 돕는다.*/

const CheckList = () => {
  // 5R 관련 각각 활동들에 대한 점수를 각각의 변수에 삽입하기 위해 초기화.
  const [number1, setNumber1] = useState(0);
  const [number2, setNumber2] = useState(0);
```

```
  const [number3, setNumber3] = useState(0);
  const [number4, setNumber4] = useState(0);
  const [number5, setNumber5] = useState(0);


  return (
    <SafeAreaView style={styles.container}>
      <Image
        source={require('../../assets/activity/countTitle.png')}
        style={{marginTop: '10%'}}
        // '5R 점수 계산하기' 라는 제목의 이미지를 상단에 제시. + marginTop: '10%' 로
상단과 여백 주기.
      />
      <View style={styles.bodyContainer}>
        <View style={styles.rowContainer}>
          {/* 점수 표기할 분야 제시(Refuse). 해당 Text는 Style에 정의되어있는
ratingLabel의 양식을 따름. */}
          <Text style={styles.ratingLabel}>Refuse</Text>
          {/* 사용자가 점수를 체크할 수 있는 표현 스타일 제시. (5개의 나뭇잎으로 0점부
터 5점까지 점수 표현할 수 있도록 제시) */}
          <Rating
          // 평가 표현 방식에 대한 디자인 설정
            type="custom"
            ratingImage={require('../../assets/activity/leaf.png')}
            ratingColor="#77B255"
            ratingBackgroundColor="#c8c7c8"
            ratingCount={5}
            imageSize={40}
            startingValue={0}
            // onFinishRating -> 사용자가 평가를 마쳤을때 정수로 최종 평가값을 제공함.
            onFinishRating={rating => {
              setNumber1(rating);
              console.log(rating);
            }}
            // paddingVertical을 이용하여 디자인 규격 설정
            style={{paddingVertical: 10}}
          />
        </View>
        <View style={styles.rowContainer}>
          {/* 점수 표기할 분야 제시(Reduce). 해당 Text는 Style에 정의되어있는
ratingLabel의 양식을 따름. */}
          <Text style={styles.ratingLabel}>Reduce</Text>
          <Rating
            type="custom"
            // eslint-disable-next-line no-undef
            ratingImage={require('../../assets/activity/leaf.png')}
            ratingColor="#77B255"
            ratingBackgroundColor="#c8c7c8"
            ratingCount={5}
            startingValue={0}
            imageSize={40}
            // Reduce에 대한 사용자의 평가값을 setNumber2에 정수로 제공받음.
            onFinishRating={rating => {
              setNumber2(rating);
            }}
            style={{paddingVertical: 10}}
          />
```

```
        </View>
        <View style={styles.rowContainer}>
          {/* 점수 표기할 분야 제시(Reuse). 해당 Text는 Style에 정의되어있는
ratingLabel의 양식을 따름. */}
          <Text style={styles.ratingLabel}>Reuse</Text>
          <Rating
            type="custom"
            // eslint-disable-next-line no-undef
            ratingImage={require('../../assets/activity/leaf.png')}
            ratingColor="#77B255"
            ratingBackgroundColor="#c8c7c8"
            ratingCount={5}
            imageSize={40}
            startingValue={0}
            // Reuse에 대한 사용자의 평가값을 setNumber3에 정수로 제공받음.
            onFinishRating={rating => {
              setNumber3(rating);
            }}
            style={{paddingVertical: 10}}
          />
        </View>
        <View style={styles.rowContainer}>
          {/* 점수 표기할 분야 제시(Recycle). 해당 Text는 Style에 정의되어있는
ratingLabel의 양식을 따름. */}
          <Text style={styles.ratingLabel}>Recycle</Text>
          <Rating
            type="custom"
            // eslint-disable-next-line no-undef
            ratingImage={require('../../assets/activity/leaf.png')}
            ratingColor="#77B255"
            ratingBackgroundColor="#c8c7c8"
            ratingCount={5}
            startingValue={0}
            // Recycle에 대한 사용자의 평가값을 setNumber4에 정수로 제공받음.
            onFinishRating={rating => {
              setNumber4(rating);
            }}
            style={{paddingVertical: 10}}
          />
        </View>
        <View style={styles.rowContainer}>
          {/* 점수 표기할 분야 제시(Rot). 해당 Text는 Style에 정의되어있는
ratingLabel의 양식을 따름. */}
          <Text style={styles.ratingLabel}>Rot</Text>
          <Rating
            type="custom"
            // eslint-disable-next-line no-undef
            ratingImage={require('../../assets/activity/leaf.png')}
            // 점수를 선택할 수 있는 나뭇잎 이미지를 불러옴.
            ratingColor="#77B255"
            ratingBackgroundColor="#c8c7c8"
            ratingCount={5}
            startingValue={0}
            imageSize={40}
            // Rot에 대한 사용자의 평가값을 setNumber5에 정수로 제공받음.
            onFinishRating={rating => {
              setNumber5(rating);
```

```
            }}
            style={{paddingVertical: 10}}
          />
        </View>
      </View>
      <View
        style={[
          styles.bodyContainer,
          {
            height: '9%',
            marginTop: -5,
            backgroundColor: '#FFE090',
          },
        ]}>
        {/* 5R 관련 사용자의 활동 체크에 대한 점수 환산 후 점수 제시 코드. 관련 Text design은 Stlye의 showText의 양식을 따름.*/}
        <Text style={[styles.showText, {color: 'black', marginTop: -1}]}>
          🔎 총 합 :{' '}
          {Math.ceil(
            // Refuse, Reuse, Rot 부분에는 가중치 2 설정. Reduce, Recycle 분야에는 가중치 3으로 설정.
            ((2 / 60) * Number(number1) +
              (3 / 60) * Number(number2) +
              (2 / 60) * Number(number3) +
              (3 / 60) * Number(number4) +
              (2 / 60) * Number(number5)) *
              100,
              // 100점 만점 설정
          )}
          점 입니다.
        </Text>
      </View>
    </SafeAreaView>
  );
};

// 해당 페이지에서 사용하는 style setting 선언.
const styles = StyleSheet.create({
  container: {
    backgroundColor: '#728F60',
    paddingHorizontal: 30,
    flex: 1,
  },
  rowContainer: {
    display: 'flex',
    flexDirection: 'row',
    justifyContent: 'space-around',
    alignItems: 'center',
  },
  ratingLabel: {
    fontSize: 20,
    fontWeight: '600',
    textAlign: 'left',
    width: 80,
  },
  headerText: {
    alignItems: 'center',
```

```
    fontSize: 30,
    color: 'white',
    fontWeight: '600',
  },
  bodyContainer: {
    marginTop: 30,
    backgroundColor: '#FDF5DC',
    paddingHorizontal: 20,
    marginVertical: 20,
    height: '60%',
    display: 'flex',
    justifyContent: 'center',
    borderRadius: 10,
  },
  textInput: {
    marginTop: 15,
    marginBottom: 15,
    paddingHorizontal: 15,
    height: 40,
    borderRadius: 10,
    borderColor: 'gray',
    borderWidth: 1,
  },
  showText: {
    marginTop: 10,
    fontSize: 18,
    color: 'white',
    fontWeight: '600',
  },
});

export default CheckList;
```

Check파트의 5R정보 탭 코드

Src/elements/activity/R5Info.js

```
/* eslint-disable no-undef */
import React from 'react';
import {StyleSheet, ScrollView, Text, View, Image} from 'react-native';
import {SafeAreaView} from 'react-native-safe-area-context';
// 이미지 Resource import

// R5Info 창에서는 R5에 속하는 활동과 그에 대한 예시와 설명을 제시하여 R5관련 기본지식
습득을 도와줌.
const R5Info = () => {
  return (
    //하단 View는 Style에 정의되어있는 container 양식을 따름.(배경)
    <SafeAreaView style={styles.container}>
      <ScrollView showsVerticalScrollIndicator={false} style={{flex: 1}}>
        {/* ScroolView를 이용하여 스크롤 할 수 있도록 설정. + false 값을 받아 스크롤을
위아래로 움직일때 가시적인 스크롤을 숨김. */}
        <Image
          source={require('../../assets/activity/countTitle.png')}
```

```
          style={{marginTop: '10%'}}
          // '5R 점수 계산하기' 라는 제목의 이미지를 상단에 제시. + marginTop: '10%'
로 상단과 여백 주기
        />

        <View style={{marginTop: '5%'}}>
          {/* 5R 관련 활동 첫번째 Refuse (거절하기) text. Style에 정의되어있는
headerText 양식을 따름. */}
          <Text style={styles.headerText}>1. Refuse</Text>
          {/* refuse 에 관련된 내용을 담기위한 틀 설정. Style에 정의되어있는
headerText 양식을 따름. */}
          <View style={styles.bodyContainer}>
            {/* 주제에 대한 간략한 설명 제시 + 해당 text는 Style에 정의되어있는
bodyTop 양식을 따름. */}
            <Text style={styles.bodyTop}>
              Say no to things you do not need, such as disposable cups and
              plastic straws.
            </Text>
            <Image
            // 하단에 제시한 상대경로의 이미지를 불러옴.
              style={{width: '90%', height: '75%'}}
              source={require('../../assets/activity/Refuse.png')}
            />
          </View>
          {/*  5R 관련 활동 두번째 Reduce (줄이기) text. Style에 정의되어있는
headerText 양식을 따름. */}
          <Text style={styles.headerText}>2. Reduce</Text>
          <View style={styles.bodyContainer}>
            <Text style={styles.bodyTop}>
              Purchasing only the essentials and buying products with little
              packaging.
            </Text>
            <Image
            // 하단에 제시한 상대경로의 이미지를 불러옴.
              style={{width: '90%', height: '75%'}}
              source={require('../../assets/activity/Reduce.png')}
            />
          </View>
          {/*  5R 관련 활동 세번째 Reuse (재사용하기) text. Style에 정의되어있는
headerText 양식을 따름. */}
          <Text style={styles.headerText}>3. Reuse</Text>
          <View style={styles.bodyContainer}>
            <Text style={styles.bodyTop}>
              Reusing something that can be reused.
            </Text>
            <Image
            // 하단에 제시한 상대경로의 이미지를 불러옴.
              style={{width: '90%', height: '75%'}}
              source={require('../../assets/activity/Reuse.png')}
            />
          </View>
          {/*  5R 관련 활동 네번째 Recycle (거절하기) text. Style에 정의되어있는
headerText 양식을 따름. */}
          <Text style={styles.headerText}>4. Recycle</Text>
          <View style={styles.bodyContainer}>
            <Text style={styles.bodyTop}>
```

```
              Recycle products that can be recycled.
            </Text>
            <Image
            // 하단에 제시한 상대경로의 이미지를 불러옴.
              style={{width: '90%', height: '75%'}}
              source={require('../../assets/activity/Recycle.png')}
            />
          </View>
          {/*  5R 관련 활동 다섯번째 Rot (썩는 제품 사용하기) text. Style에 정의되어있는 headerText 양식을 따름. */}
          <Text style={styles.headerText}>5. Rot</Text>
          <View style={styles.bodyContainer}>
            <Text style={styles.bodyTop}>
              In the case of food that cannot be recycled, collect only food and rot to make feed.
            </Text>
            <Image
            // 하단에 제시한 상대경로의 이미지를 불러옴.
              style={{height: '65%', resizeMode: 'contain'}}
              source={require('../../assets/activity/Rot.png')}
            />
          </View>
        </View>
      </ScrollView>
    </SafeAreaView>
  );
};

// 해당 페이지에서 사용하는 style setting 선언.
const styles = StyleSheet.create({
  container: {
    backgroundColor: '#728F60',
    paddingHorizontal: 30,
    height: 100,
    flex: 1,
  },
  headerText: {
    fontSize: 30,
    color: 'white',
    fontWeight: '600',
  },
  bodyContainer: {
    marginTop: 30,
    backgroundColor: '#FDF5DC',
    paddingHorizontal: 20,
    marginVertical: 20,
    height: 220,
    borderRadius: 10,
    alignItems: 'center',
    justifyContent: 'space-evenly',
  },
  bodyTop: {
    fontSize: 15,
  },
});
```

```
export default R5Info;
```

3. Information 파트 코드

정보페이지 전체 컴포넌트를 배치

src/components/Information.js

```
import React from 'react';
import {
  StyleSheet,
  Image,
  Dimensions,
  ScrollView,
  Platform,
} from 'react-native';
import {SafeAreaView} from 'react-native-safe-area-context';
import News from '../elements/Information/News';
import mainLogo from '../assets/information/logo.png';
import Mypage from '../elements/Information/MyPage';

// 정보 페이지 전체 컴포넌트를 리턴
const Information = () => {
  return (
    <SafeAreaView style={styles.container}>
      {/* 스크롤이 가능한 레이아웃 배치 */}
      <ScrollView style={{marginTop: '-5%'}}>
        <Image
          source={mainLogo}
          // style={{resizeMode: 'cover', transform: [{scale: 0.67}]}}
          style={{
            resizeMode: 'contain',
            height: 160,
            width: '100%',
            marginBottom: '3%',
          }}
        />
        {/* elements 에서 정의한 mypage, news 컴포넌트 배치 */}
        <Mypage />
        <News />
      </ScrollView>
    </SafeAreaView>
  );
};


// layout 스타일 정의, 정렬 방식, 가로, 세로길이, padding 등
const styles = StyleSheet.create({
  container: {
```

```
    display: 'flex',
    width: Dimensions.get('screen').width,
    height: Platform.OS == 'ios' ? '105%' : '100%',
    alignItems: 'center',
    justifyContent: 'flex-start',
    paddingTop: 10,
  },
});

export default Information;
```

src/elements/Information/MyPage.js

```
import React from 'react';
import {Text, Dimensions} from 'react-native';
import {LineChart} from 'react-native-chart-kit';
//linechart를 이용해 그래프를 그린다
/* 기록 된 5R 점수들을 날짜별로 비교하여 보기 용이하게 하기 위해 그래프 사용 */


const Mypage = () => {
  return (
    <>
      <Text
      //linechart 제목 디자인
        style={{
          fontSize: 20,
          fontWeight: '600',
          alignSelf: 'flex-start',
          marginLeft: '5%',
          marginBottom: '5%',
        }}>
        🌱 나의 기록 확인하기
      </Text>
      <LineChart
        data={{
          labels: ['05/16', '05/17', '05/18', '05/19', '05/20', '05/21'],
          datasets: [
            {
              data: [80, 89, 90, 98, 100, 92],
            },
          ],/* 그래프 안의 기본 데이터 설정 */
        }}
        width={Dimensions.get('window').width} // linechart 디자인
        height={250}
```

```
        yAxisInterval={1}
        chartConfig={{
          backgroundColor: '#8F6D82',
          backgroundGradientFrom: '#8F6D82',
          backgroundGradientTo: '#8F6D82',
          decimalPlaces: 2,
          color: (opacity = 1) => `rgba(255, 255, 255, ${opacity})`,
          labelColor: (opacity = 1) => `rgba(255, 255, 255, ${opacity})`,
          style: {
            borderRadius: 16,
          },
          propsForDots: {
            r: '6',
            strokeWidth: '2',
            stroke: 'white',
          },
        }}
        bezier
        style={{
          paddingTop: 30,
        }}
      />
    </>
  );
};


export default Mypage;
```

실시간 제로웨이스트 뉴스를 받아오는 News

src/elements/Information/News.js

```
import axios from 'axios';
import React, {useState} from 'react';
import {Linking, StyleSheet, Text, View} from 'react-native';
import {Config} from 'react-native-config';
import Slick from 'react-native-slick';

// Naver Developers 에 등록한 애플리케이션 id, password 개발 환경 파일에서 가져오기
const CLIENT_ID = Config.CLIENT_ID;
const CLIENT_SECRET = Config.CLIENT_SECRET;

// News api 호출을 위한 request 데이터 정의
// url - json 값을 반환하도록 요청 url 정의
const url = 'https://openapi.naver.com/v1/search/news.json';
```

```
// headers 세팅
const headers = {
  'X-Naver-Client-Id': CLIENT_ID,
  'X-Naver-Client-Secret': CLIENT_SECRET,
};
// request parameter 정의, 검색 키워드 : 제로웨이스트, 5 개 기사 크롤링, 검색시작 위치
1, 날짜 기준 정렬
const params = {query: '제로웨이스트', display: 5, start: 1, sort: 'date'};

// api 로 부터 response 를 받기 위한 axios 통신
// 위에서 정의한 url, headers, params 를 넣어 get 방식의 통신을 진행한다.
const ecoNews = axios
  .get(url, {
    params: params,
    headers,
  })
  .then(res => {
    return res.data.items;
  })
  .catch(err => console.log(err));

// 화면 UI 그리기
const News = () => {
  // 받아온 news 를 저장할 변수 선언
  const [newsList, setNews] = useState(null);

  // promise 형태의 비동기 데이터를 처리해서 newsList 에 저장
  ecoNews.then(res => {
    setNews(res);
  });

  // UI 를 리턴한다.
  return (
    // 뉴스 페이지 전체를 감싸는 Layout
    <View style={styles.container}>
      <Text
        style={{
          fontSize: 20,
          fontWeight: '600',
          alignSelf: 'flex-start',
          marginLeft: '5%',
          marginBottom: '5%',
          color: 'black',
        }}>
        🌏 매일 녹색 소식
      </Text>
```

```
      {/* newsList 에 있는 데이터를 Slick-slider 로 보여준다.
          뉴스의 배경색을 두가지 색깔을 번갈아가면서 띄운다.
          사용성을 고려하여, 넘김 버튼을 배치하였으며 직접 슬라이드로 넘기기도 가능하다.
          하단 페이지네이션은 가독성을 위하여 제거하였다.
      */}
      <Slick showsButtons={true} showsPagination={false}>
        {newsList ? (
          newsList.map((data, idx) => (
            <View
              style={idx % 2 == 1 ? slickStyles.slide1 : slickStyles.slide2}
              key={idx}>
              {/* 제목 및 본문에 특수문자와 html 요소를 제거해준다. */}
              <Text
                style={slickStyles.newsTitle}
                onPress={() => {
                  Linking.openURL(data.link);
                }}>
                {data.title
                  .replace(/<[^>]*>?/gm, '')
                  .replace(/&quot;|&amp;/g, '')}
              </Text>
              <Text style={slickStyles.newsText}>
                {data.description
                  .replace(/<[^>]*>?/gm, '')
                  .replace(/&quot;/g, '')}
              </Text>
              {/* 뉴스 발행 날짜 중 불필요한 형식을 제거한다. */}
              <Text style={slickStyles.newsDate}>
                {data.pubDate.replace('+0900', '')}
              </Text>
            </View>
          ))
        ) : (
          // 통신에 실패했을 경우를 위한 대체 UI 를 배치한다.
          <View style={[slickStyles.slide1, {alignItems: 'center'}]}>
            <Text style={slickStyles.newsLodingText}>뉴스 로딩 중...</Text>
          </View>
        )}
      </Slick>
    </View>
  );
};

// StyleSheet 를 활용한 스타일 정의
// Layout 구조, 글꼴, 색, padding, margin 등을 사전에 지정한다.
const styles = StyleSheet.create({
  container: {
```

```
    marginTop: '10%',
    height: 300,
    display: 'flex',
    alignItems: 'center',
    justifyContent: 'flex-start',
  },
});

const slickStyles = StyleSheet.create({
  slide1: {
    flex: 1,
    padding: '10%',
    justifyContent: 'center',
    backgroundColor: '#A98D8D',
  },
  slide2: {
    flex: 1,
    padding: '10%',
    justifyContent: 'center',
    backgroundColor: '#92BBC0',
  },
  newsTitle: {
    color: '#fff',
    fontSize: 21,
    fontWeight: 'bold',
  },
  newsDate: {
    marginTop: '3%',
    color: '#fff',
    fontSize: 13,
    alignSelf: 'flex-end',
  },
  newsText: {
    marginTop: '5%',
    color: '#fff',
    fontSize: 15,
  },
  newsLodingText: {color: '#fff', fontSize: 21, fontWeight: 'bold'},
});

export default News;
```