캡스톤 디자인

10.26 진도 발표

발표자 : 김가희

팀장 : 김현성

팀원 : 김가희, 정영훈

 # Bicycle Application

자전거 네비게이션

자전거 네비게이션 기능을 탑재한

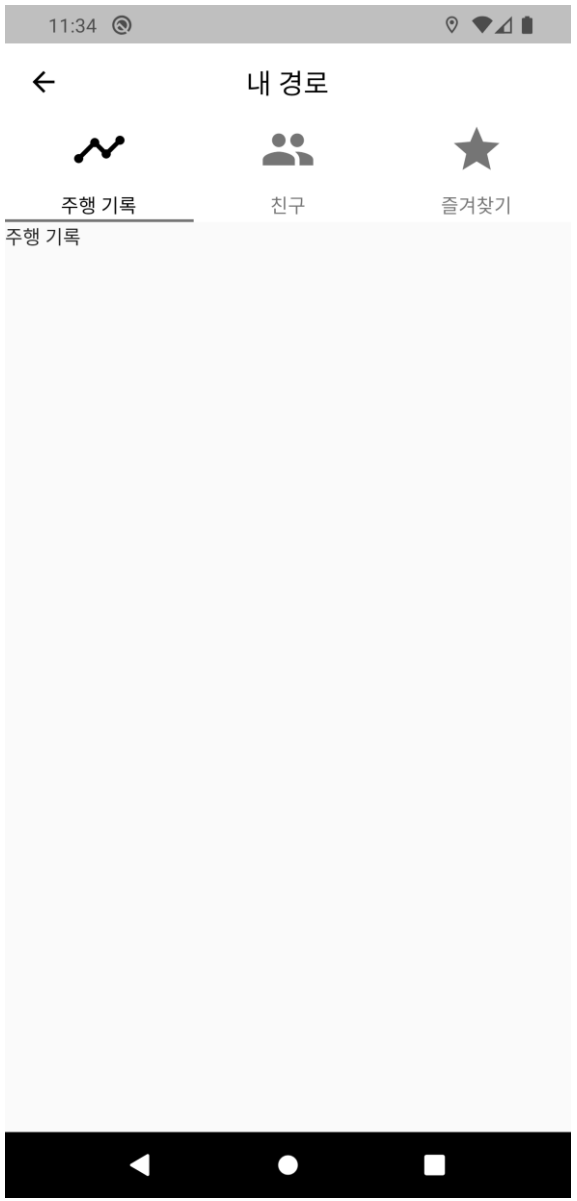
자전거 전용 애플리케이션

 # Bicycle Application

목차

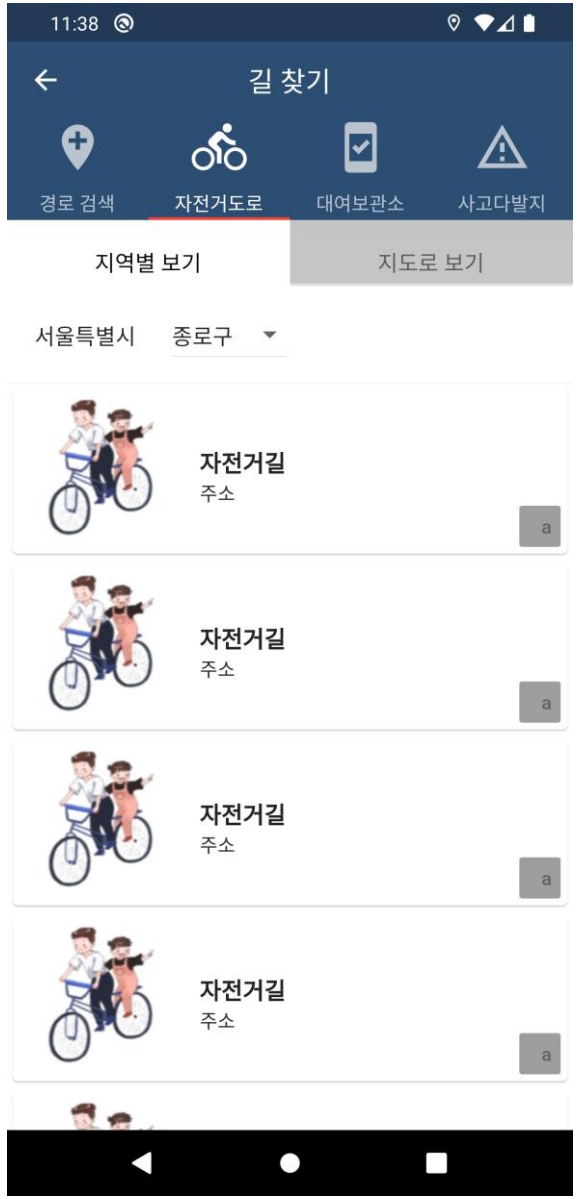
UI

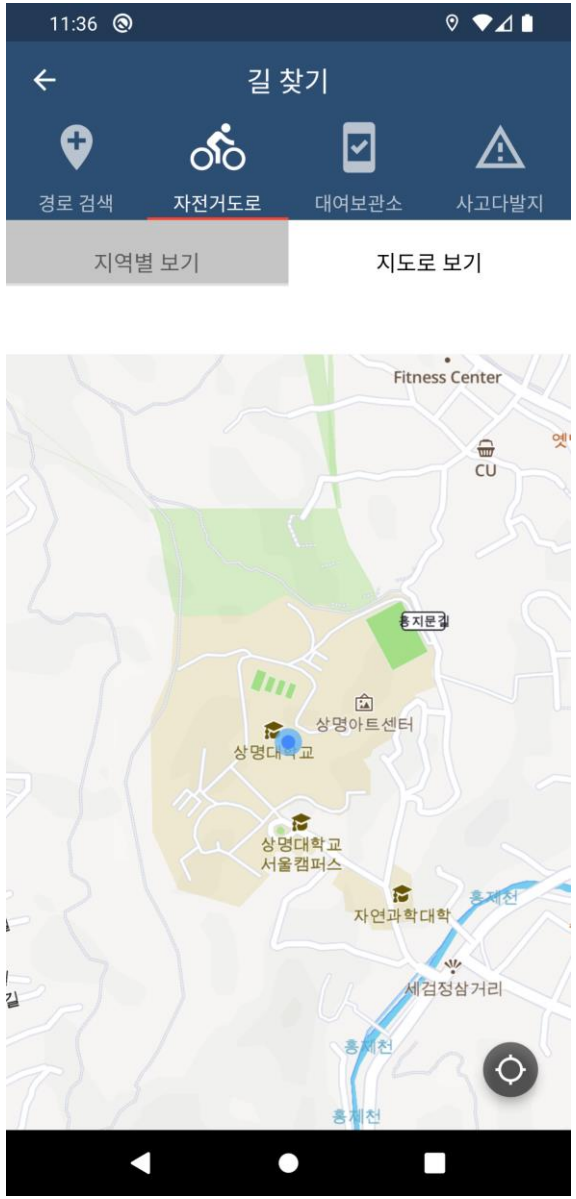

11:33
자전거 내비게이션
길 찾기
내 경로
구기터널
구기터널
구기동
구기터널
비봉길
평창길
GS 슈퍼마켓
신영동
상명대학교
BBQ
홍지동
자하문로
홍지문터널
홍지문터널
서울미술
계단
내 정보
설정


11:34
내 경로
주행 기록
친구
즐겨찾기
주행 기록


11:34
길 찾기
경로 검색
자전거도로
대여보관소
사고다발지
지역별 보기
지도로 보기
서울특별시
종로구
강남구
영등포구
마포구
자전거길
주소
자전거길
주소
자전거길
주소

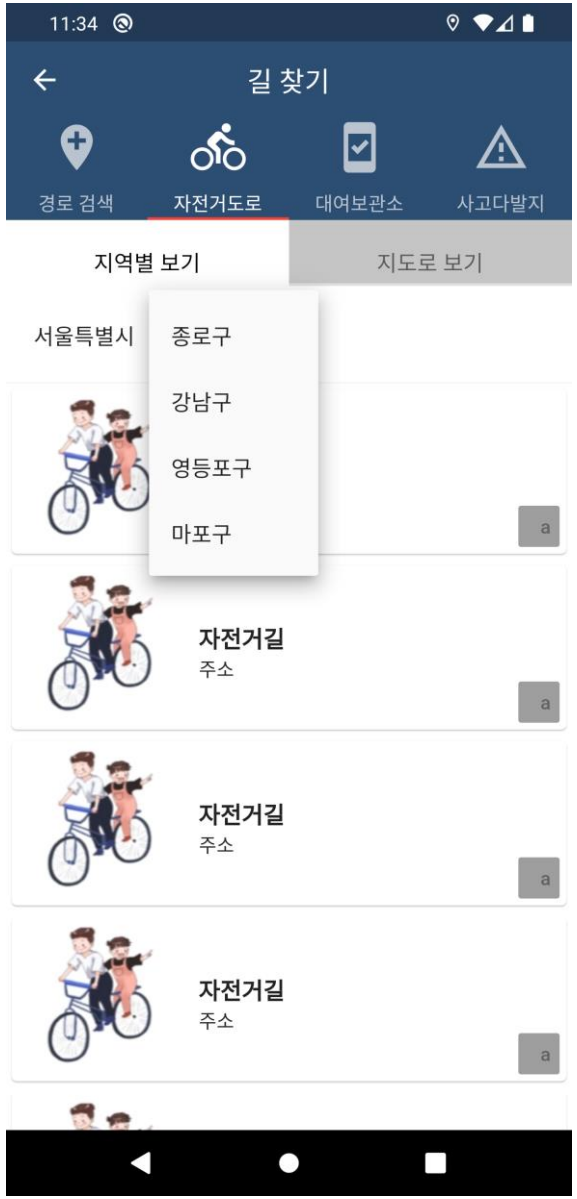
자전거 도로 UI





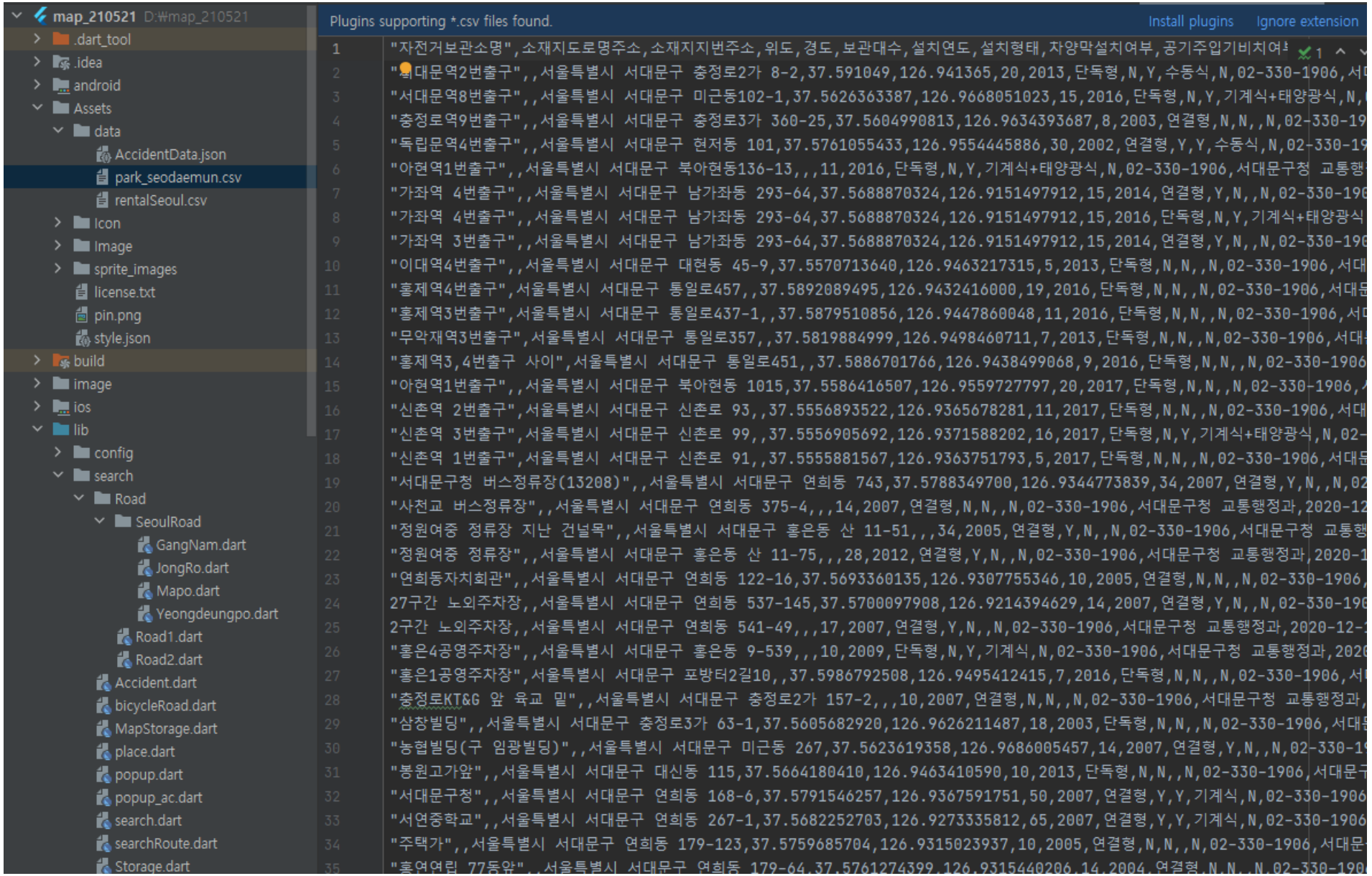

```
children: <Widget>[
  Material(
    color: Colors.black12,
    child: TabBar(
      unselectedLabelColor: Colors.black54,
      labelColor: Colors.black,
      indicatorColor: Colors.white,
      controller: _tabController,
      labelPadding: const EdgeInsets.all(0.0),
      tabs: [
        _getTab(0,
            Text("\n지역별 보기",
              textAlign: TextAlign.center,
              style: TextStyle(fontSize: 16.0),)),  // Text
        _getTab(1,
            Text("\n지도로 보기",
              textAlign: TextAlign.center,
              style: TextStyle(fontSize: 16.0),)),  // Text
      ],
      onTap: _onTap,
    ),  // TabBar
  ),  // Material
  Expanded(
    child: TabBarView(
      physics: NeverScrollableScrollPhysics(),
      controller: _tabController,
      children: [
        Road1(),
        Road2(),
      ],
```

자전거 도로 세부 메뉴

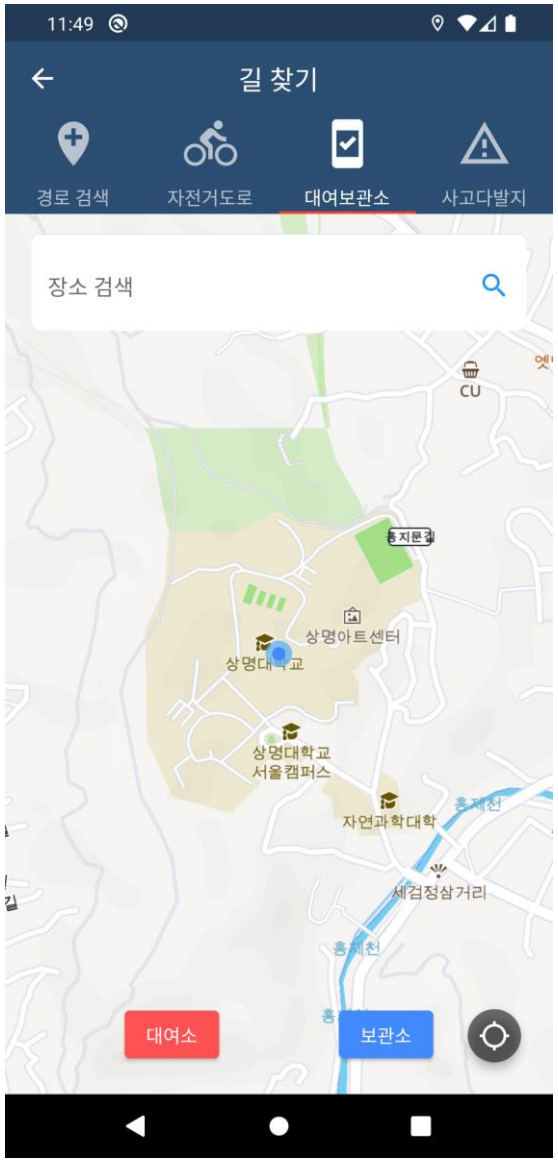


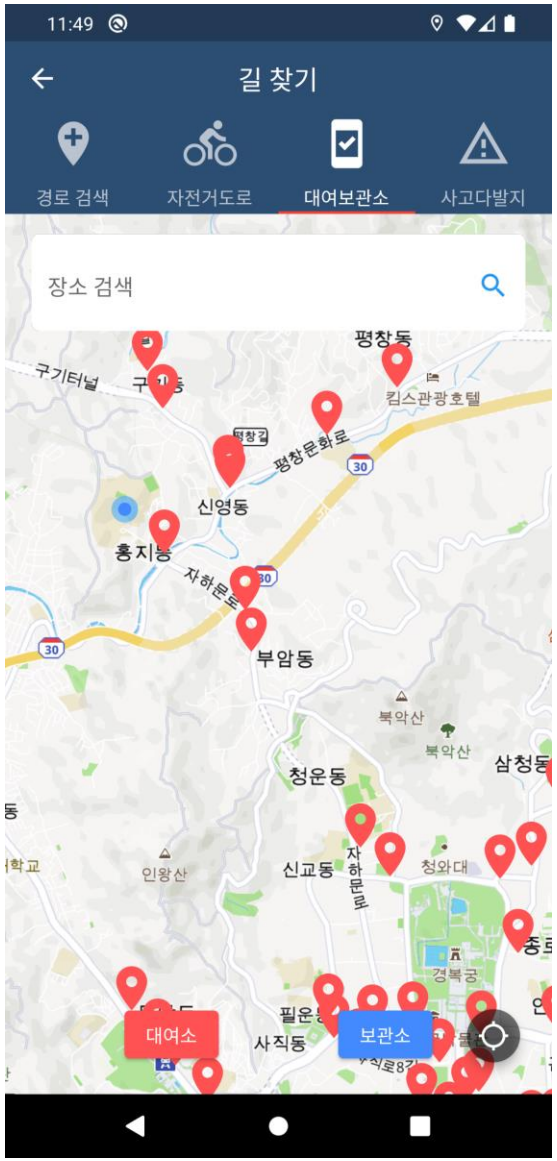

```dart
Container(
  child: DropdownButton(
    value: _selectedArea,
    items: AreaList.map(
          (value) {
        return DropdownMenuItem(
          value: value,
          child: Text(value),
        );  // DropdownMenuItem
      },
    ).toList(),
    onChanged: (value) {
      setState(() {
        _selectedArea = value as String;
        switch (_selectedArea) {
          case '종로구':
            selectedPage = 0;
            break;
          case '강남구':
            selectedPage = 1;
            break;
          case '영등포구':
            selectedPage = 2;
            break;
          case '마포구':
            selectedPage = 3;
            break;
        }
        _pageController.jumpToPage(selectedPage);
        _tabController = TabController(length: _AreaList.length, vsync: this,
            initialIndex: selectedPage);  // TabController
      });
    },
  ),  // DropdownButton
```

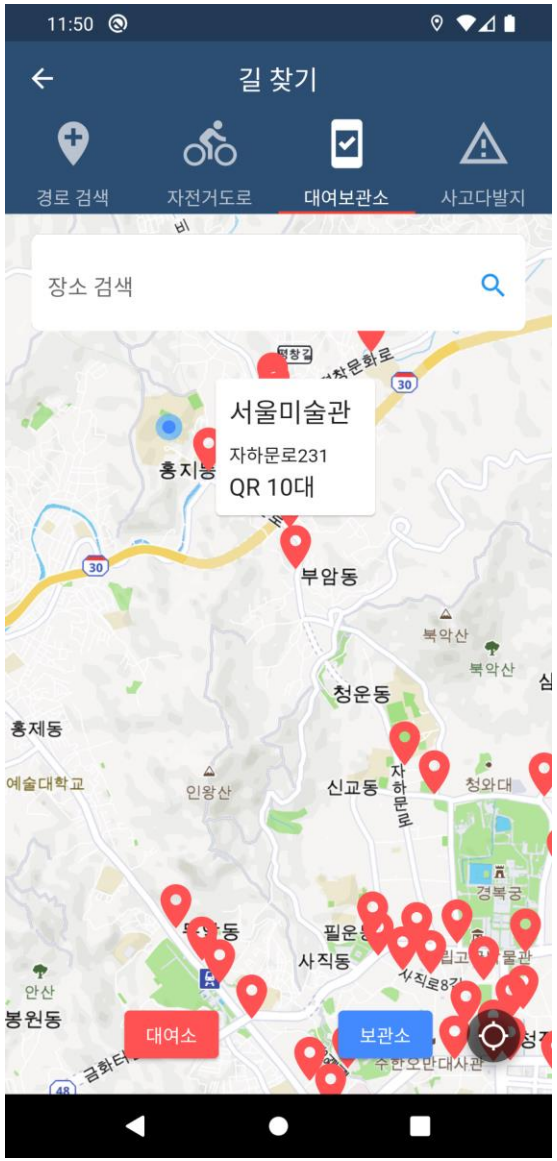
대여소, 보관소 CSV 데이터

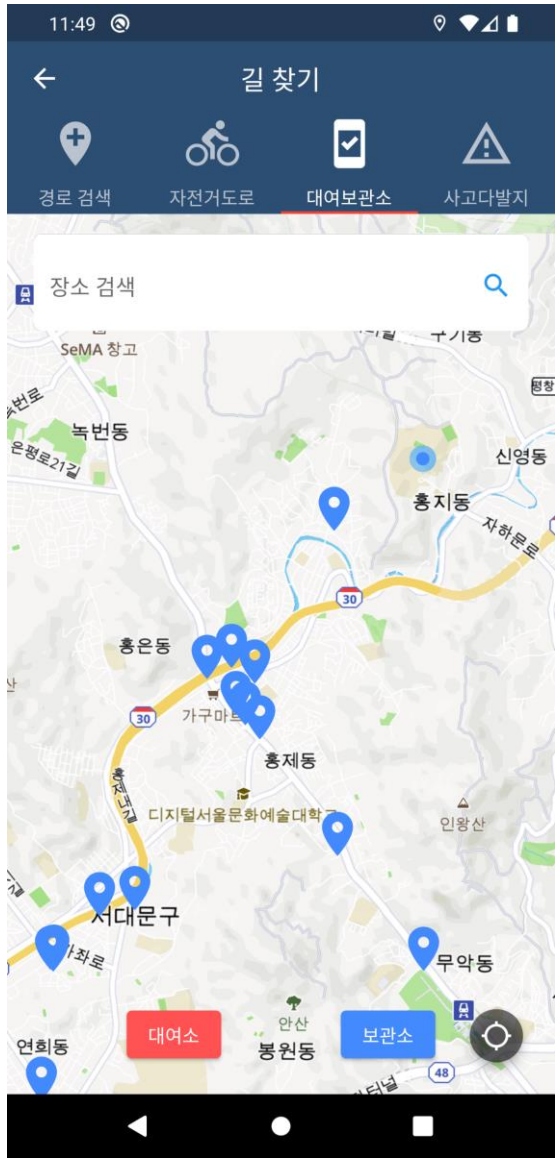


대여소, 보관소 표시

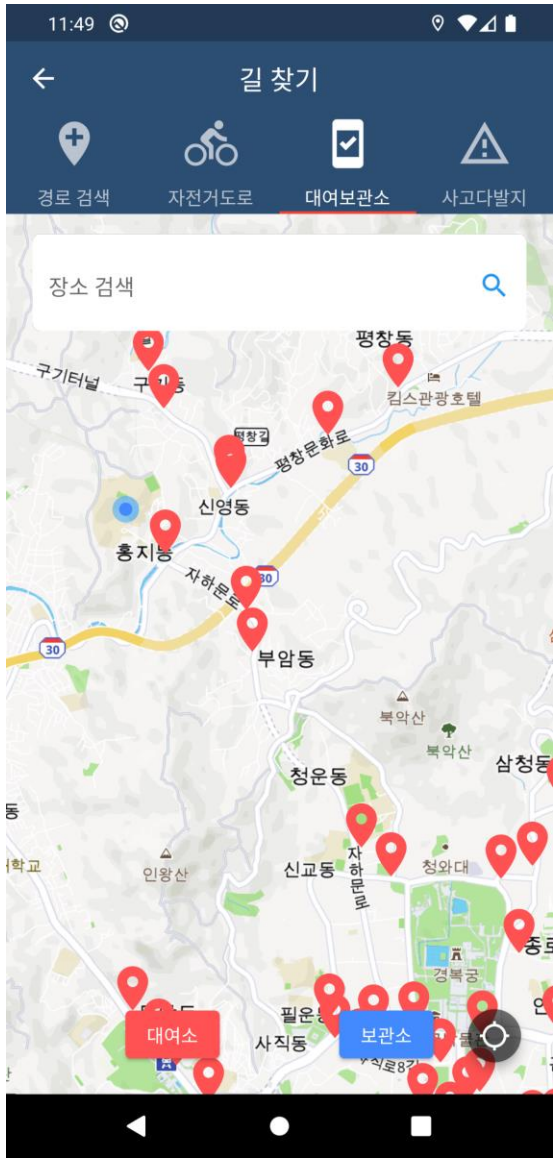

11:49
길 찾기
경로 검색
자전거도로
대여보관소
사고다발지
장소 검색
CU
상명아트센터
상명대학교
상명대학교
서울캠퍼스
자연과학대학
세검정삼거리
대여소
보관소


11:49
길 찾기
경로 검색
자전거도로
대여보관소
사고다발지
장소 검색
평창동
구기터널
킴스관광호텔
평창문화로
신영동
홍지동
자하문로
부암동
북악산
청운동
삼청동
인왕산
신교동
청와대
경복궁
대여소
보관소


11:50
길 찾기
경로 검색
자전거도로
대여보관소
사고다발지
장소 검색
서울미술관
자하문로231
QR 10대
홍지동
부암동
북악산
청운동
홍제동
예술대학교
인왕산
신교동
청와대
경복궁
사직동
봉원동
대여소
보관소


11:49
길 찾기
경로 검색
자전거도로
대여보관소
사고다발지
장소 검색
SeMA 창고
녹번동
신영동
홍지동
자하문로
홍은동
가구마을
홍제동
디지털서울문화예술대학교
인왕산
서대문구
무악동
연희동
안산
봉원동
대여소
보관소

대여소, 보관소 표시



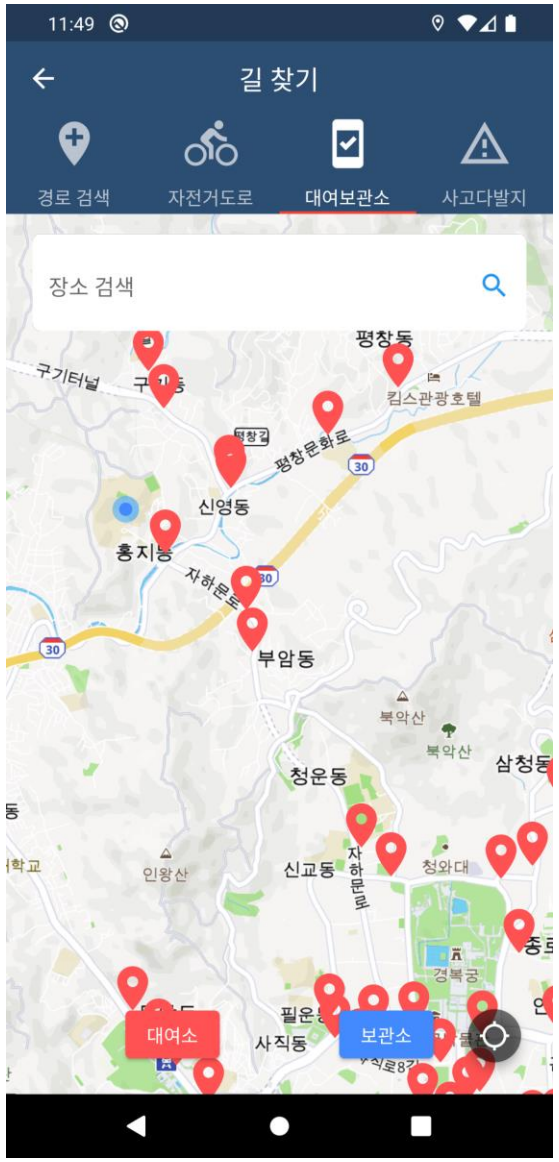

```
List<LatLng> _markerPositions = [];
List<String> _markerAddress = [];
List<List<dynamic>> csvdata = [];
```

- 위도 경도 값을 저장할 LatLng 리스트
- 해당 위치 값의 주소를 저장할 String 리스트
- 대여보관소 데이터를 받아올 이중 dynamic 타입 리스트

```
loadCSV_rental() async {
  final Data = await rootBundle.loadString('Assets/data/rentalSeoul.csv');
  List<List<dynamic>> csvData = CsvToListConverter().convert(Data);
  setState(() {
    csvdata = csvData;
    for (int i=0; i<2170; i++){
      if (csvdata[i][2] == '종로구'){
        double la = csvdata[i][4];
        double lo = csvdata[i][5];
        _markerPositions.add(LatLng(la, lo));
        _markerAddress.add(csvdata[i][1]);
      }
    }
    rental = true;
  });
}
```

- CsvToListConverter 함수로 csv 데이터를 변환하여 이중 리스트에 저장
- 반복문으로 이중 리스트의 4, 5 index에 입력되어있는 double 타입의 위도, 경도 값을 LatLng 타입으로 변환 후 리스트에 저장
- 같은 방식으로 주소값 문자열 리스트 저장
- 대여소 버튼을 누를때마다 rental = true;로 설정하여 보관소와 구분

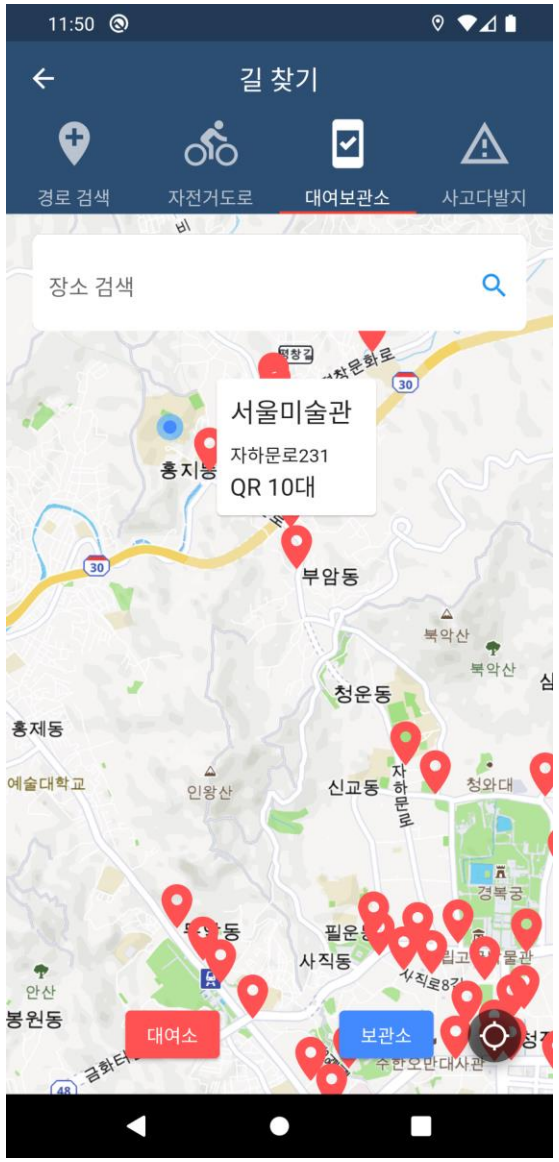
대여소, 보관소 표시




```dart
Container(
  child: ElevatedButton(
    onPressed: () async {
      _popupLayerController.hideAllPopups();
      setState(() {
        _markerPositions.clear();
      });
      await loadCSV_rental();
      print(rental);
    },
    style: ButtonStyle(
      backgroundColor: MaterialStateProperty.all(Colors.redAccent),
    ),  // ButtonStyle
    child: Text("대여소"),
  ),  // ElevatedButton
),  // Container
Spacer(),
Container(
  child: ElevatedButton(
    onPressed: () async {
      _popupLayerController.hideAllPopups();
      setState(() {
        _markerPositions.clear();
      });
      await loadCSV_park();
      print(rental);
    },
    style: ButtonStyle(
      backgroundColor: MaterialStateProperty.all(Colors.blueAccent),
    ),  // ButtonStyle
    child: Text("보관소"),
  ),  // ElevatedButton
),  // Container
```

버튼을 누를때마다
_markerPositions.clear();
이전에 표시됐던 marker
리스트를 모두 삭제한 후
대여소 또는 보관소 함수 호출

대여소, 보관소 정보 팝업



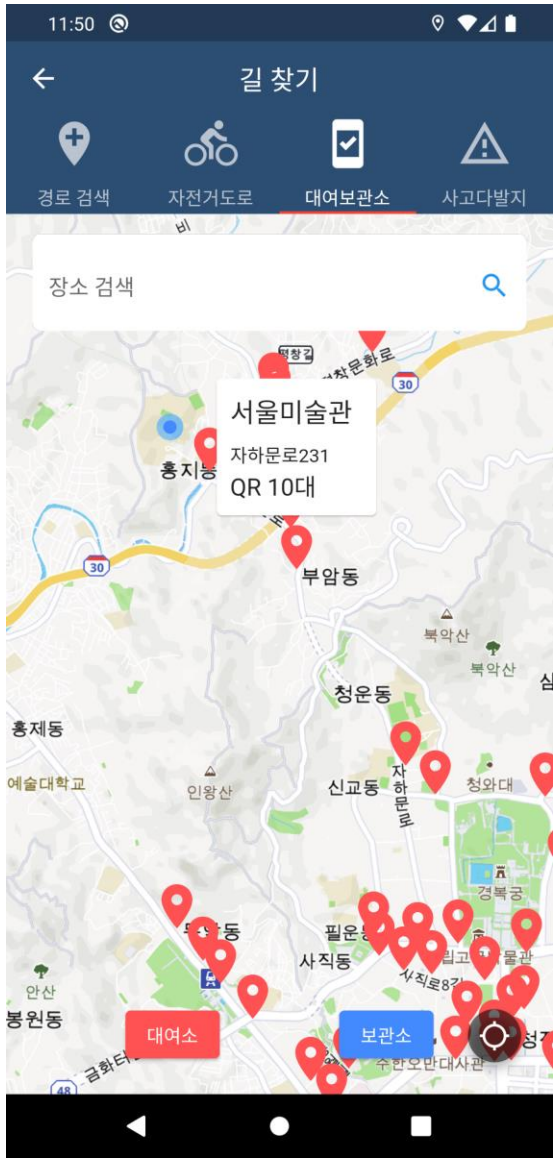

```
List<Marker> get _markers => _markerPositions
    .map(
      (markerPosition) => Marker(
    point: markerPosition,
    width: 40,
    height: 40,
    builder: (_) => Icon(Icons.location_on, size: 40, color:
    (rental == true ? Colors.redAccent : Colors.blueAccent),),
    anchorPos: AnchorPos.align(AnchorAlign.top),
  ),  // Marker
).toList();
```

- 위치 값이 저장된 리스트에서 point값을 가져와 marker 생성
- rental = true 일 경우와 아닐 경우 아이콘의 색이 바뀜
- 생성한 marker를 _markers라는 리스트로 변환 => .toList();

```
PopupMarkerLayerWidget(
  options: PopupMarkerLayerOptions(
    popupController: _popupLayerController,
    markers: _markers,
    popupAnimation: PopupAnimation.fade(duration: Duration(milliseconds: 700)),
    markerRotateAlignment:
    PopupMarkerLayerOptions.rotationAlignmentFor(AnchorAlign.top),
    popupBuilder: (BuildContext context, Marker marker) =>
        Popup(marker),
  ),  // PopupMarkerLayerOptions
),  // PopupMarkerLayerWidget
```

- 팝업 마커 위젯 패키지 이용

대여소, 보관소 정보 팝업 위젯




```dart
String Addr = '';
String LCDQR = '';
String AddrF = '';
List<List<dynamic>> csvdata = [];

@override
void initState() {
  super.initState();
  _loadCSV();
}
```

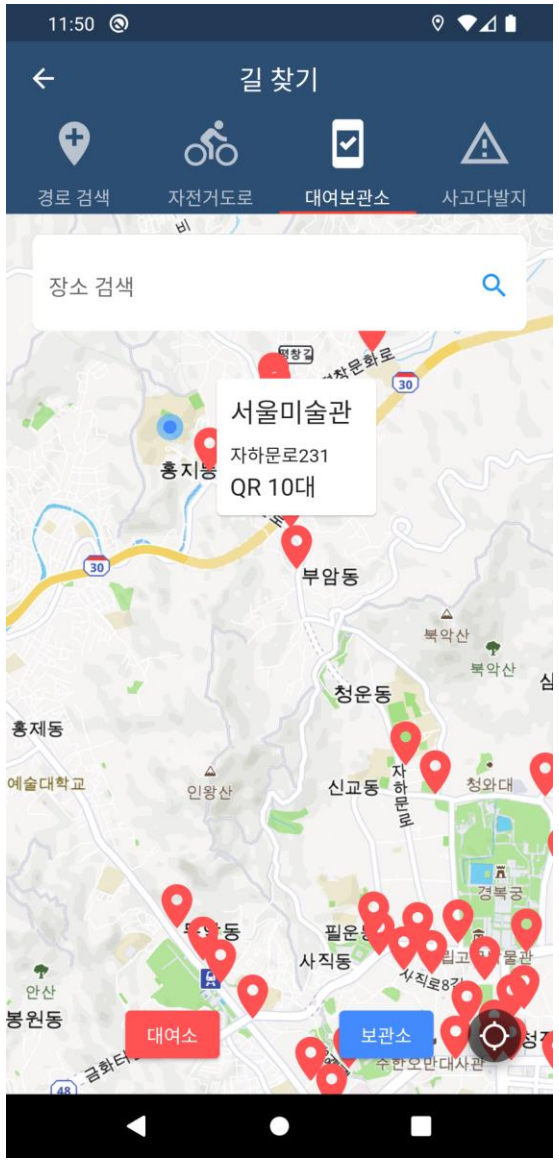
- 대여, 보관소 이름
- LCD / QR 여부
- 주소

- 자동으로 _loadCSV 함수 호출

```dart
late String path;
_loadCSV() async {
  if (rental == true){
    path = 'Assets/data/rentalSeoul.csv';
  }
  else if (rental == false){
    path = 'Assets/data/park_seodaemun.csv';
  }
  final Data = await rootBundle.loadString(path);
  List<List<dynamic>> csvData = CsvToListConverter().convert(Data);
```

- 대여소 버튼을 눌렀을 때 (rental == true) 대여소 데이터 링크 호출
- 보관소 버튼을 눌렀을 때 (rental == false) 보관소 데이터 링크 호출하여 convert

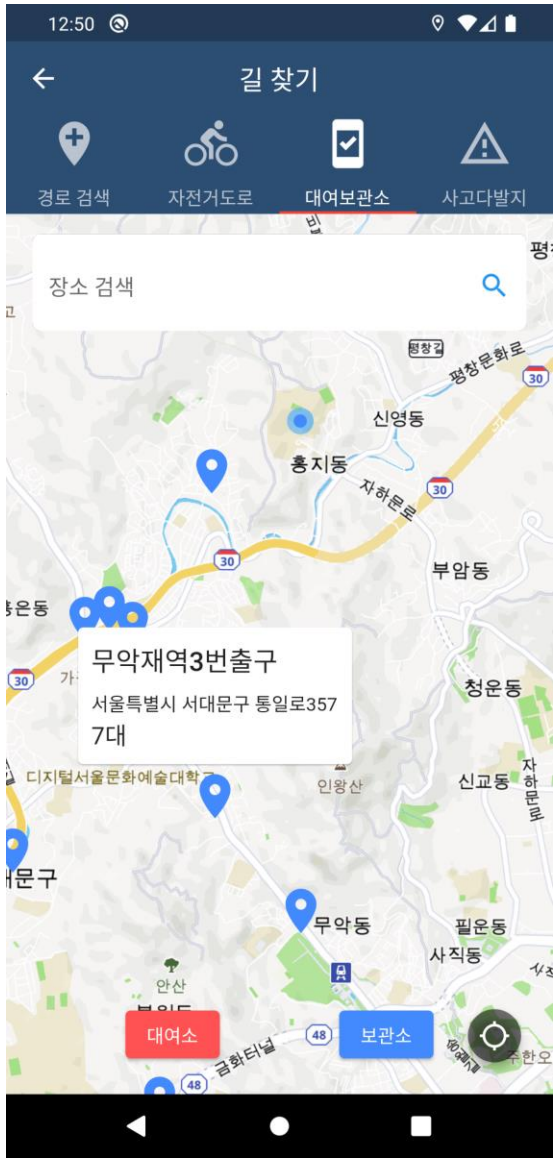
대여소, 보관소 정보 팝업 위젯




```
if (rental == true){
  for (var i=0; i<2170; i++){
    if (_marker.point.latitude == csvdata[i][4] && _marker.point.longitude == csvdata[i][5]){
      Addr = csvdata[i][1];
      if (csvdata[i][9] == 'LCD'){
        LCDQR = "LCD " + csvdata[i][7].toString() + "대";
      }
      else {
        LCDQR = "QR " + csvdata[i][8].toString() + "대";
      }
      AddrF = csvdata[i][3];
    }
  }
}
else if (rental == false){
  for(var i=1; i<51; i++){
    if (_marker.point.latitude == csvdata[i][3] && _marker.point.longitude == csvdata[i][4]){
      Addr = csvdata[i][0];
      LCDQR = csvdata[i][5].toString() + "대";
      AddrF = csvdata[i][1];
    }
  }
}
```

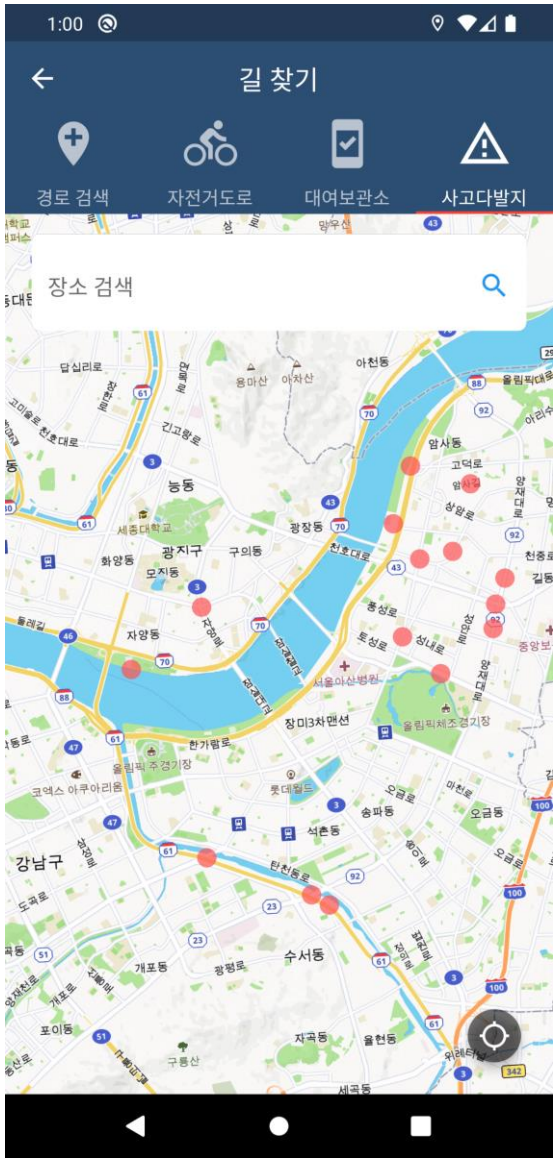
터치한 marker point의 위치값과 일치하는 대여, 보관소 데이터를 반복문을 이용하여 찾고 해당 주소와 정보들을 각각의 변수에 저장

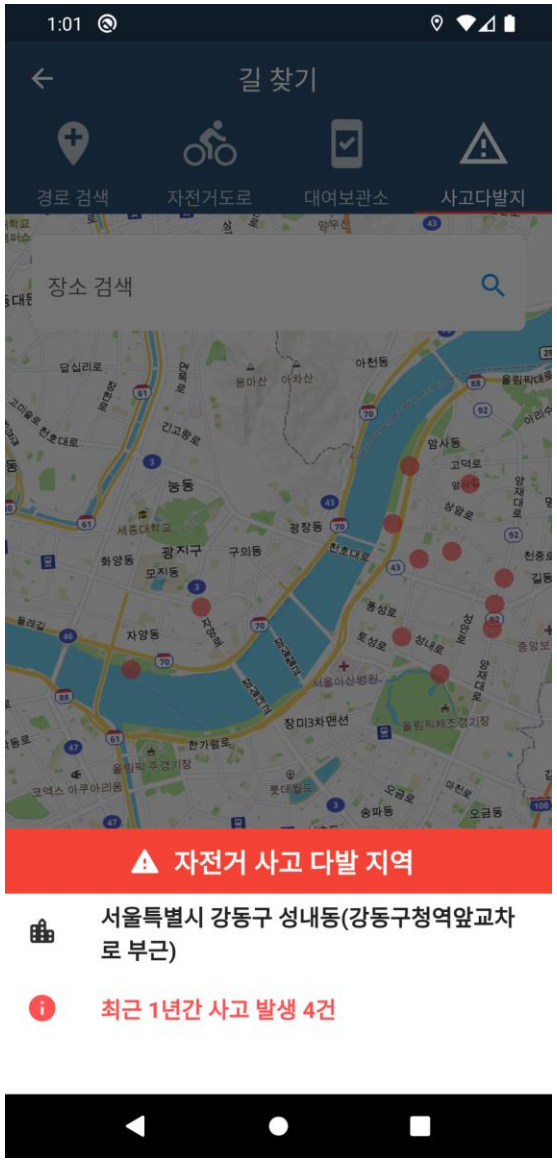
대여소, 보관소 정보 팝업 위젯



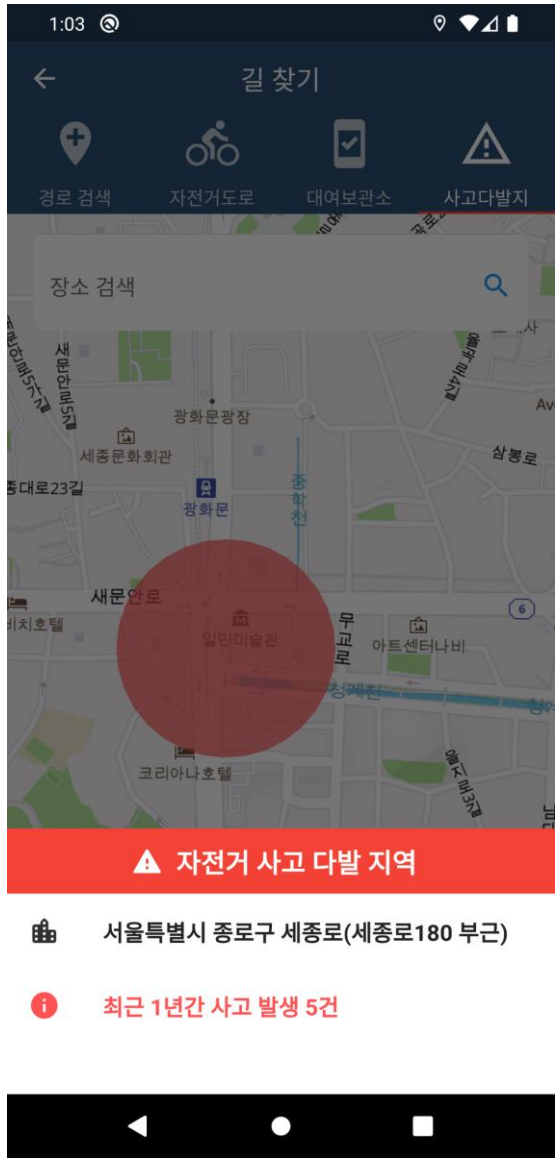

```dart
Widget _cardDescription(BuildContext context) {
  return Padding(
    padding: const EdgeInsets.all(10),
    child: Container(
      constraints: BoxConstraints(minWidth: 100, maxWidth: 300),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        mainAxisAlignment: MainAxisAlignment.start,
        mainAxisSize: MainAxisSize.min,
        children: <Widget>[
          Text(
            Addr,
            overflow: TextOverflow.fade,
            softWrap: false,
            style: const TextStyle(
              fontWeight: FontWeight.w500,
              fontSize: 20.0,
            ),  // TextStyle
          ),  // Text
          const Padding(padding: EdgeInsets.symmetric(vertical: 4.0)),
          Text(
            AddrF,
            style: const TextStyle(fontSize: 14.0),
          ),  // Text
          Text(
            LCDQR,
            style: const TextStyle(fontSize: 18.0),
          ),  // Text
```

사고 다발 지역 표시

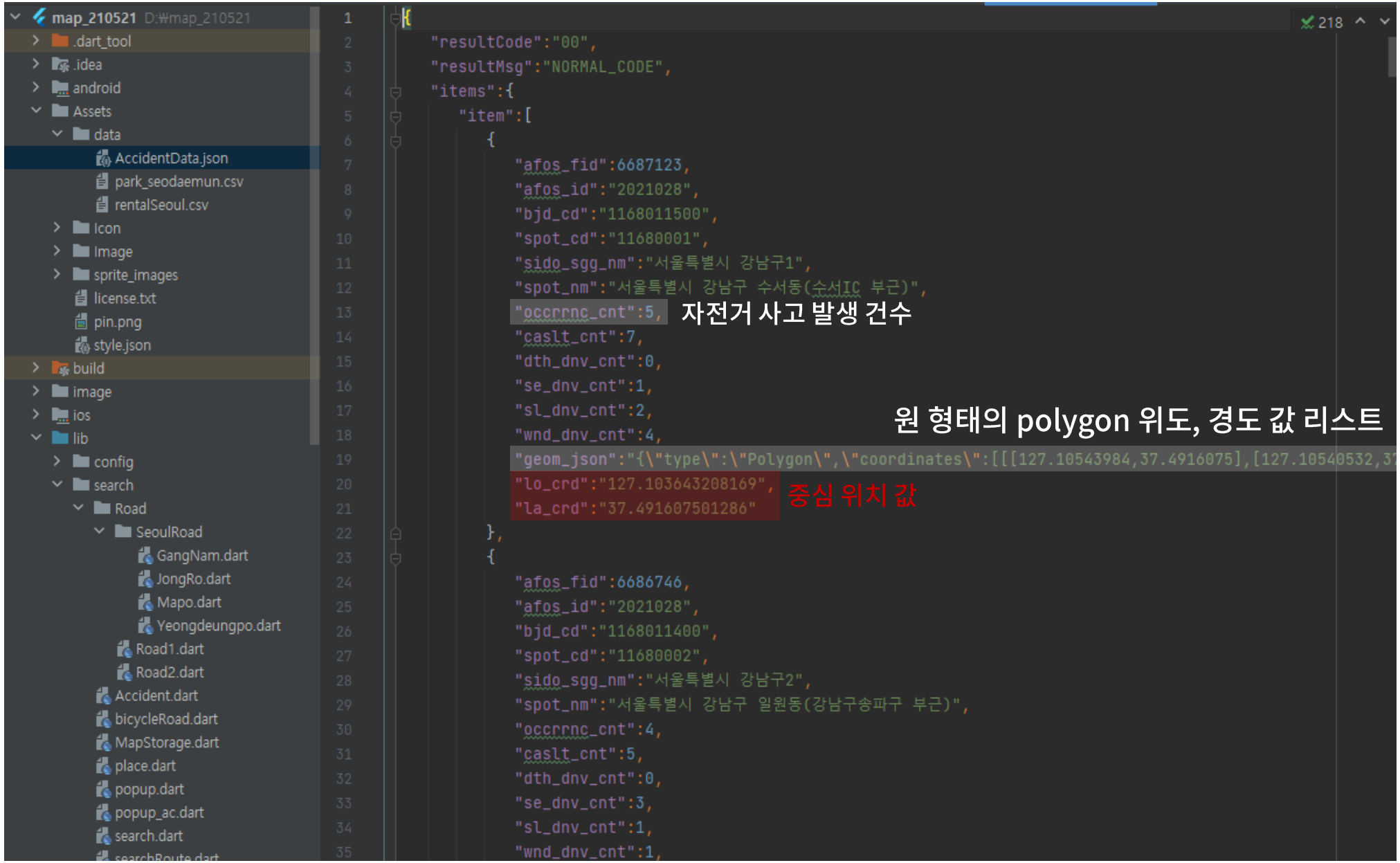

1:00
길 찾기
경로 검색
자전거도로
대여보관소
사고다발지
장소 검색


1:01
길 찾기
경로 검색
자전거도로
대여보관소
사고다발지
장소 검색
자전거 사고 다발 지역
서울특별시 강동구 성내동(강동구청역앞교차로 부근)
최근 1년간 사고 발생 4건


1:03
길 찾기
경로 검색
자전거도로
대여보관소
사고다발지
장소 검색


1:03
길 찾기
경로 검색
자전거도로
대여보관소
사고다발지
장소 검색
자전거 사고 다발 지역
서울특별시 종로구 세종로(세종로180 부근)
최근 1년간 사고 발생 5건

사고 다발 지역 JSON 데이터



사고 다발 지역 표시

```
String path = 'Assets/data/AccidentData.json';
loadJsonData() async {
  final String response = await rootBundle.loadString(path);
  Map<String, dynamic> acd = jsonDecode(response);
  String jsonData;
  String jsonData_la, jsonData_lo;
  String jsonData_add;
  int jsonData_cnt;
  int num = acd['totalCount'];
```

① JSON 데이터를 String 형태 response로 불러온 뒤 Map<String, dynamic> 이라는 타입의 변수에 jsonDecode 함수를 이용해 decode 하여 저장

② 반복문을 이용해서 json 데이터의 key 값에 맞는 value 값들을 각각 타입에 맞는 리스트들에 저장

③ 하나의 지역이 원 모양으로 표시되고
지도에 여러 개의 지역을 표시해야 하기 때문에
하나의 원을 만드는 25개 위도 경도값들의 리스트를
이중 리스트에 저장

④ 이중 리스트에 저장된 point로 30개 polygon 생성

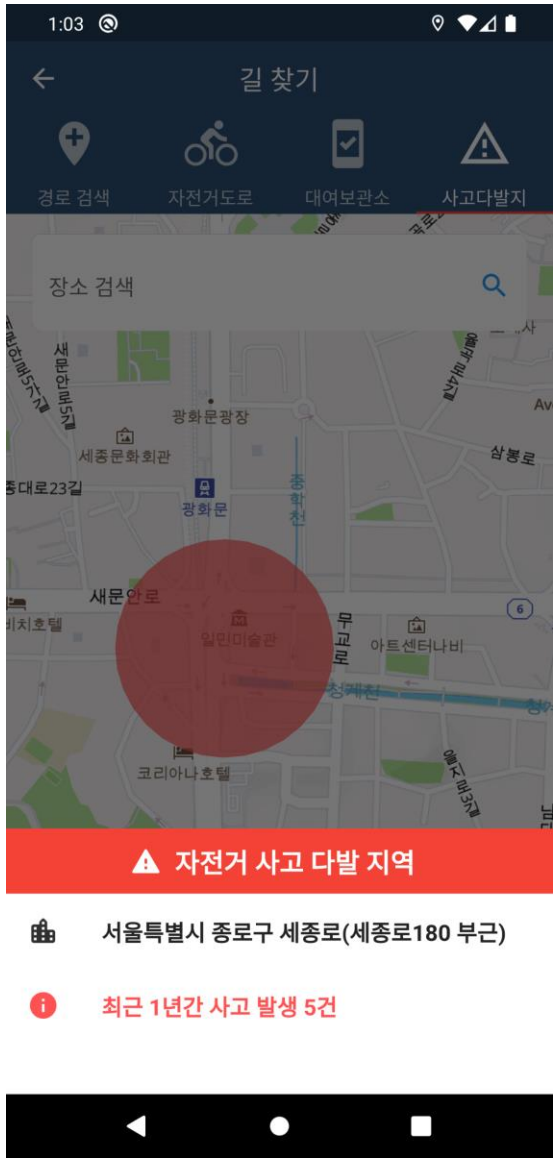
```
for (i = 0; i < 30; i++) {
  jsonData_la = acd['items']['item'][i]['la_crd'];
  jsonData_lo = acd['items']['item'][i]['lo_crd'];
  _la = double.parse(jsonData_la);
  _lo = double.parse(jsonData_lo);
  centerPoints.add(LatLng(_la, _lo)); // 중심값

  jsonData_add = acd['items']['item'][i]['spot_nm'];
  _Address.add(jsonData_add); // 주소값

  jsonData_cnt = acd['items']['item'][i]['occrrnc_cnt'];
  _Occr.add(jsonData_cnt); // 발생건수

  jsonData = acd['items']['item'][i]['geom_json'];
  temp = jsonDecode(jsonData)['coordinates'][0];
  areaPoints.add(temp);
  for (j = 0; j < temp.length; j++) {
    latlng = LatLng(areaPoints[i][j][1], areaPoints[i][j][0]);
    _areaPoints.add(latlng);
  }
  areaList.add(_areaPoints.toList());
  _areaPoints.clear();
}
polygon = [Polygon(points: areaList[0], color: Colors.redAccent.withOpacity(0.7))];
for(k=1; k<30; k++){
  polygon.add(
    Polygon(points: areaList[k],
        color: Colors.redAccent.withOpacity(0.7)),
  );
}
```

사고 다발 지역 정보 팝업


```
List<Marker> get _markers => centerPoints
    .map(
        (markerPosition) => Marker(
        width: 40.0,
        height: 40.0,
        point: markerPosition,
        builder: (ctx) =>
        new Container(
            child: IconButton(
              icon: Icon(Icons.brightness_1_outlined),
              color: Colors.transparent,
              iconSize: 0.0,
              onPressed: () {
                _showModalBottomSheet(markerPosition);
              },
            )  // IconButton
        )  // Container
    )  // Marker
).toList();
```

기본적인 polygon에는 onPressed 기능 X
mapbox 지도에 사용 가능한 관련 패키지도 존재하지 않음
=> polygon의 중심 위치값에 투명 marker 추가하여
사용자가 해당 부분 터치 시 위치 정보 제공

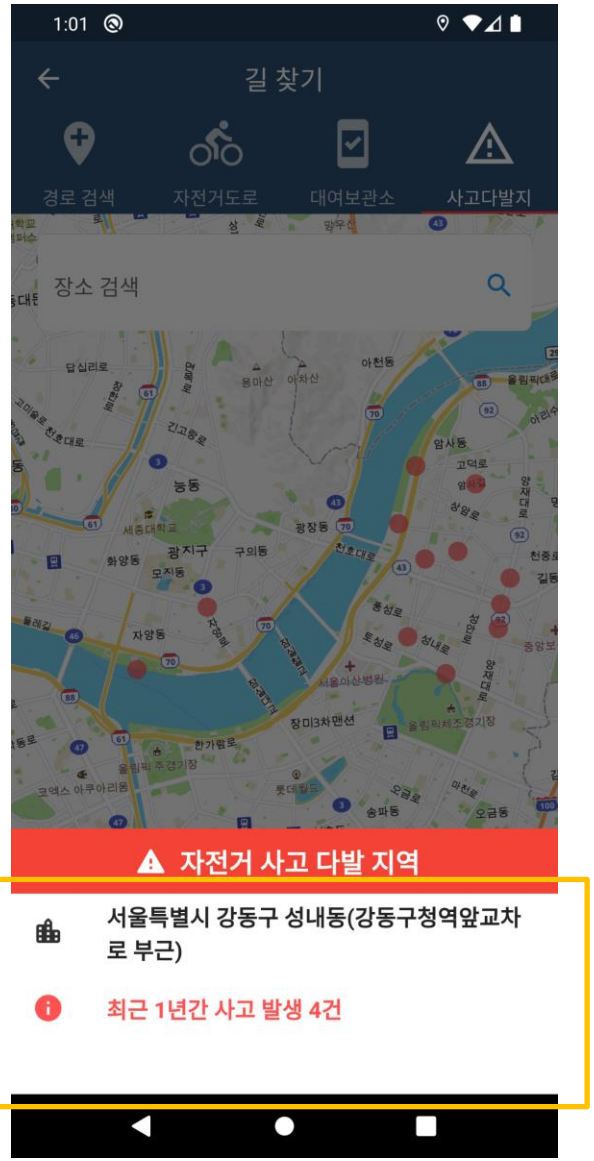
사고 다발 지역 정보 팝업




```
void _showModalBottomSheet(LatLng latlng){
  for (var p=0; p<30; p++){
    if(latlng == centerPoints[p]){
      Addr = _Address[p];
      Occr = _Occr[p];
    }
  }
  showModalBottomSheet(
      context: context,
      builder: (builder) {
        return Container(
          width: double.infinity,
          color: Colors.white,
          height: 200.0,
          child: Column(
            children: <Widget>[
              Container(
                width: double.infinity,
                color: Colors.red,
                child: Row(
```

터치한 marker가 사고 다발 지역 위치와 일치할 경우
해당 위치의 주소값과 사고 발생 건수를
showModalBottomSheet 위젯을 이용하여 출력

사고 다발 지역 정보 팝업




```dart
Align(
  alignment: Alignment.center,
  child: Column(
    children: <Widget>[
      Container(
        child: ListTile(
          leading: Icon(Icons.location_city_rounded,
            color: Colors.grey[850],
          ),  // Icon
          title: Text(Addr,
            style: TextStyle(fontSize: 17.0,
                fontWeight: FontWeight.w700),),  // TextStyle, Text
          onTap: (){
            Navigator.pop(context);
          },
        ),  // ListTile
      ),  // Container
      Container(
        child: ListTile(
          leading: Icon(Icons.info,
            color: Colors.redAccent,
          ),  // Icon
          title:
          Text('최근 1년간 사고 발생 ' + Occr.toString() + '건',
              style: TextStyle(fontSize: 17.0, color: Colors.redAccent,
                  fontWeight: FontWeight.w700)),  // TextStyle, Text
          onTap: (){
            Navigator.pop(context);
          },
```

THANK YOU