

캡스톤 디자인 11.23 진도 발표

발표자 : 김가희

팀장 : 김현성

팀원 : 김가희, 정영훈

Bicycle Application

자전거 네비게이션

자전거 네비게이션 기능을 탑재한

자전거 전용 애플리케이션



Bicycle Application

목차

01

FIRESTORE

02

친구 기능

Firestore – RealTime Database

capstone ▾

Cloud Firestore

데이터 규칙 색인 사용량

🏠 > users > CH0bTNrsNwN...


capstone-b3aa1	users	CH0bTNrsNwN8VYmSSJ4Iwyib7FA2
+ 컬렉션 시작	+ 문서 추가	+ 컬렉션 시작
users >	CH0bTNrsNwN8VYmSSJ4Iwyib7FA2 >	bookmarks friends records
	P3LeS3rKJKVngak2iUZdz8wjXGz1 TcIKbjd0hwao3pMXvVnADedhFIi2 ULD7bAuTnXSp0M2xDPN4BHzgjxL2 eKFVcYdTw3PB4lfzC4HawB3cU0V2 upIptGjfigYiv5A4MUWjeWo7zzF2	+ 필드 추가 location: [37.6026367° N, 126.9552517° E] name: "test4@test.com" public: "true" uid: "CH0bTNrsNwN8VYmSSJ4Iwyib7FA2"

Cloud Firestore 위치: nam5 (us-central)

Firestore – RealTime Database

```
class DatabaseService {  
    final String? uid;  
    DatabaseService({this.uid});  
    final CollectionReference Collection = FirebaseFirestore.instance.collection('users');  
  
    Future updateUserData(String name, String userid, GeoPoint location) async {  
        await Collection.doc(uid).set({  
            'name' : name,  
            'uid' : userid,  
            'location' : GeoPoint(curr_lat, curr_lng),  
            'public' : true,  
        });  
        await Collection.doc(uid).collection('friends').doc(uid).set({  
            'location' : GeoPoint(curr_lat, curr_lng),  
            'name' : name,  
            'uid' : uid  
        });  
        await Collection.doc(uid).collection('bookmarks').doc(uid).set({  
        });  
        await Collection.doc(uid).collection('records').doc(uid).set({  
            'Origin' : GeoPoint(0, 0),  
            'Destination' : GeoPoint(0, 0),  
        });  
    }  
}
```

회원가입과 동시에 이메일, uid, 위치 정보,
친구, 주행 기록 폴더 등 파이어스토어에 해당 회원의 데이터 컬렉션 자동 생성

 capstone-b3aa1

+ 컬렉션 시작

users

 TcIKbjd0hwao3pMXvVnADedhFli2

+ 컬렉션 시작

bookmarks

friends

records

+ 필드 추가

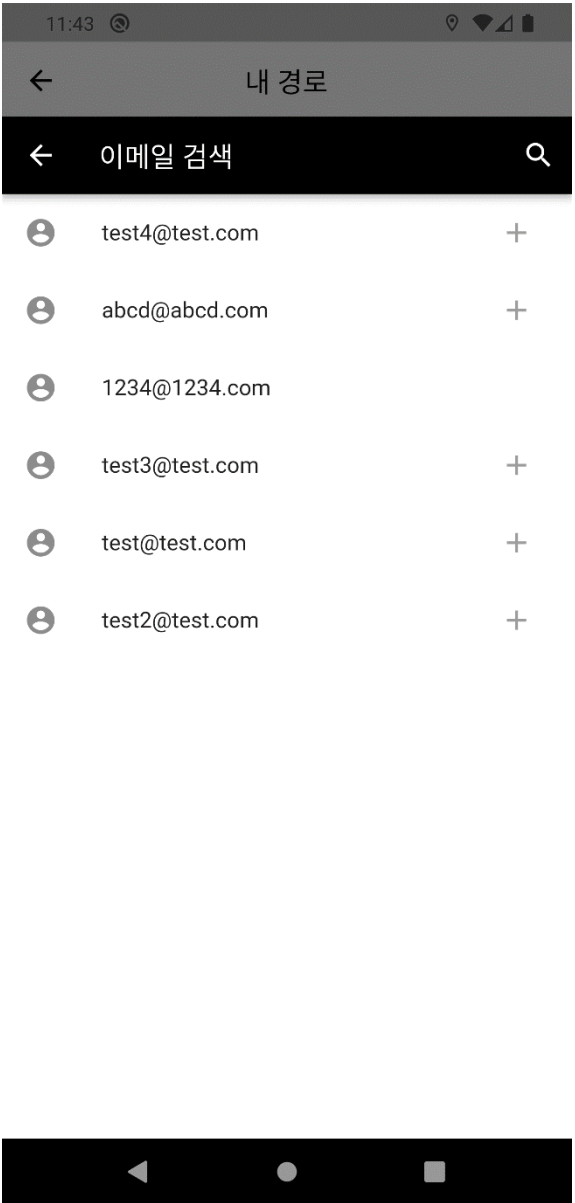
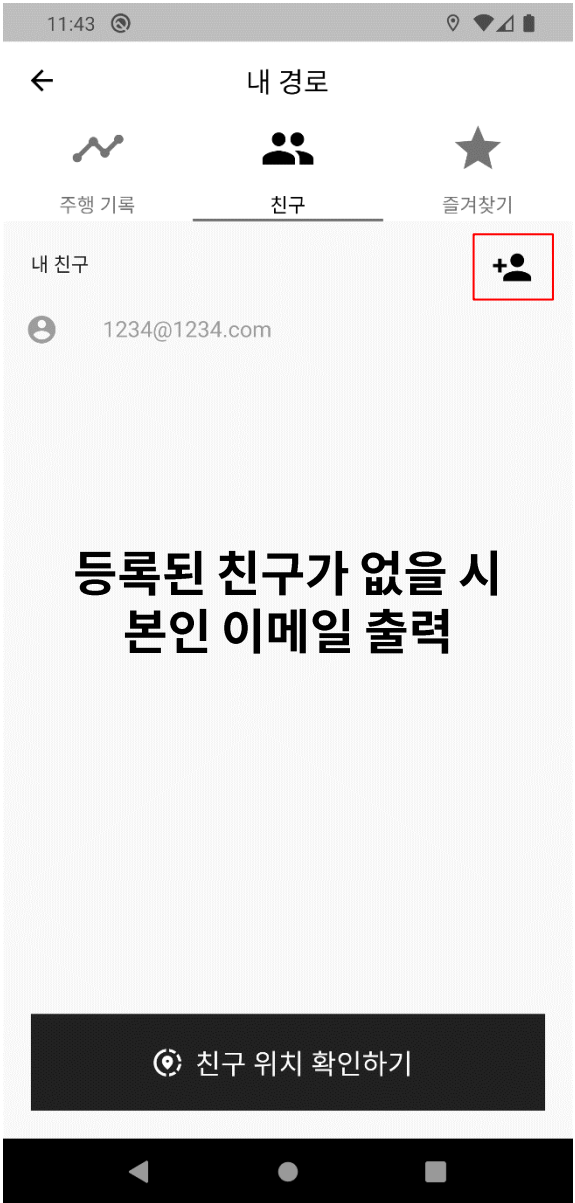
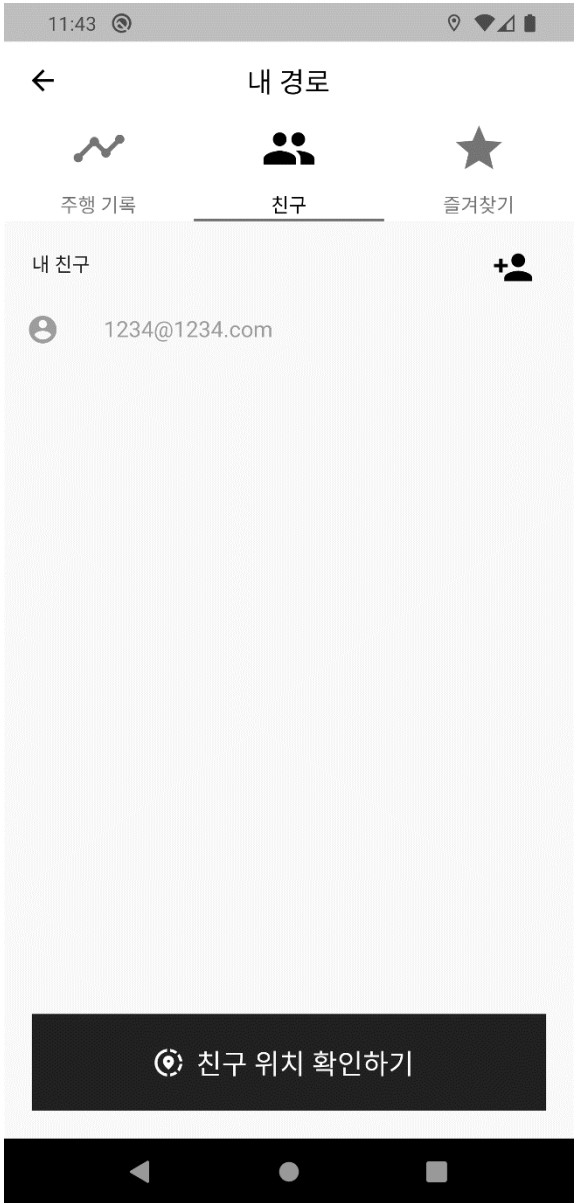
location: [37.6026367° N, 126.9552517° E]

name: "1234@1234.com"

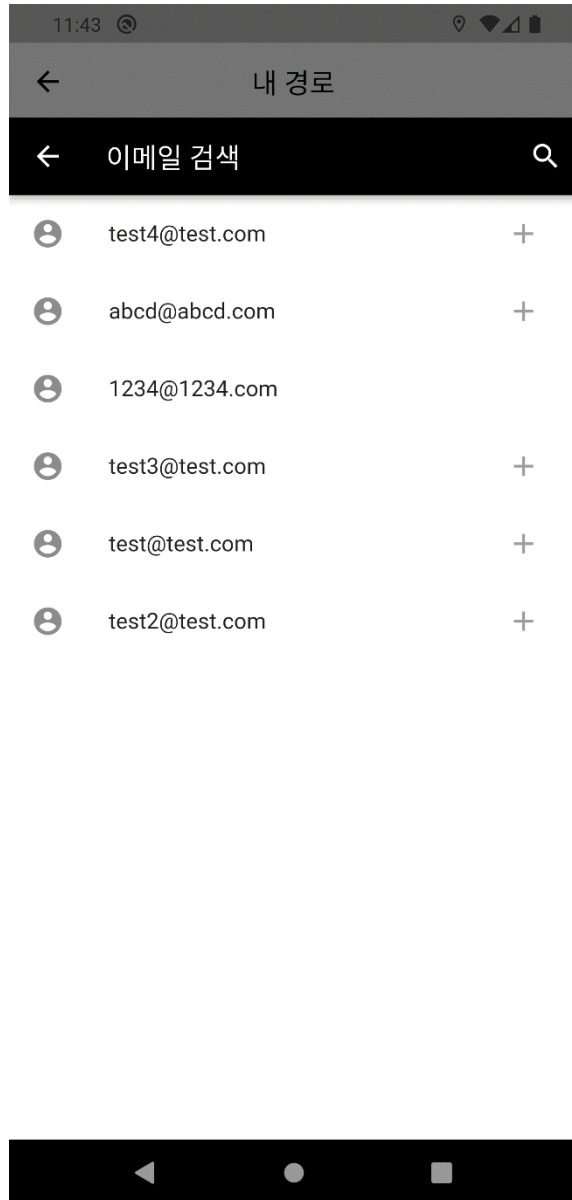
public: true

uid: "TcIKbjd0hwao3pMXvVnADedhFli2"

친구 기능



친구 기능

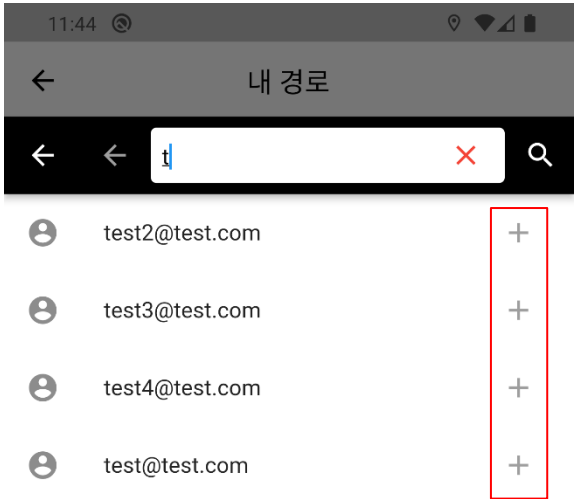


```
@override
Widget build(BuildContext context) {
  return StreamBuilder<QuerySnapshot>(
    stream: _firestore.collection('users').snapshots(),
    builder: (context, snapshot) {
      if (!snapshot.hasData){
        return Center(
          child: CircularProgressIndicator(),
        ); // Center
      }
      names.clear();
      locs.clear();
      vids.clear();
      final users = snapshot.data!.docs;
      for(var user in users){
        var username = user['name'];
        var uid = user['uid'];
        GeoPoint geoPoint = user['location'];
        locs.add(LatLng(geoPoint.latitude, geoPoint.longitude));
        names.add(username.toString());
        vids.add(uid.toString());
      }
      for(int pos=0; pos<names.length; pos++){
        if(vids[pos] == currUser())
          currUsername = names[pos];
      }
      print(currUsername);
      return FirestoreSearchScaffold(
        firestoreCollectionName: 'users',
        searchBy: 'name',
```

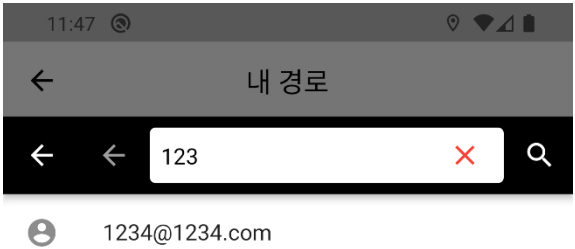
StreamBuilder<QuerySnapshot>
코드는 Firestore의 데이터를
CRUD 할 수 있도록 실시간으로 받아옴

-> 사용자들의 정보를 가져와
List에 데이터별로 저장하여
검색할 수 있도록 출력

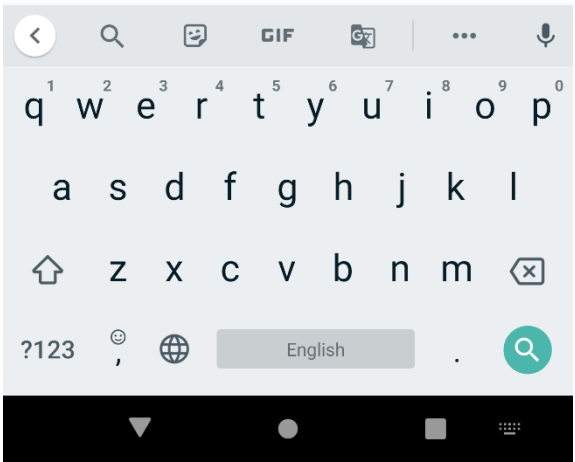
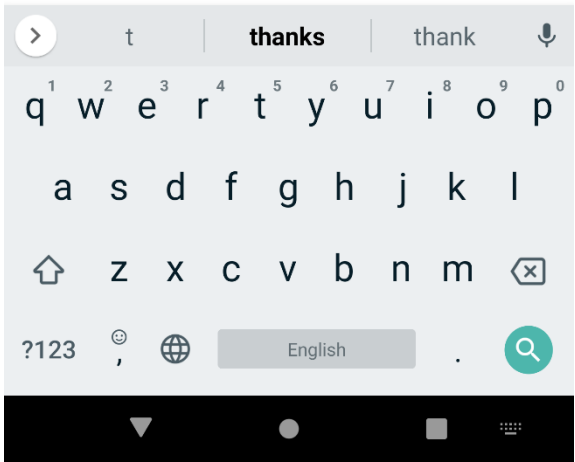
친구 추가



친구 추가 버튼



로그인중인 계정에는 버튼이 뜨지 않음



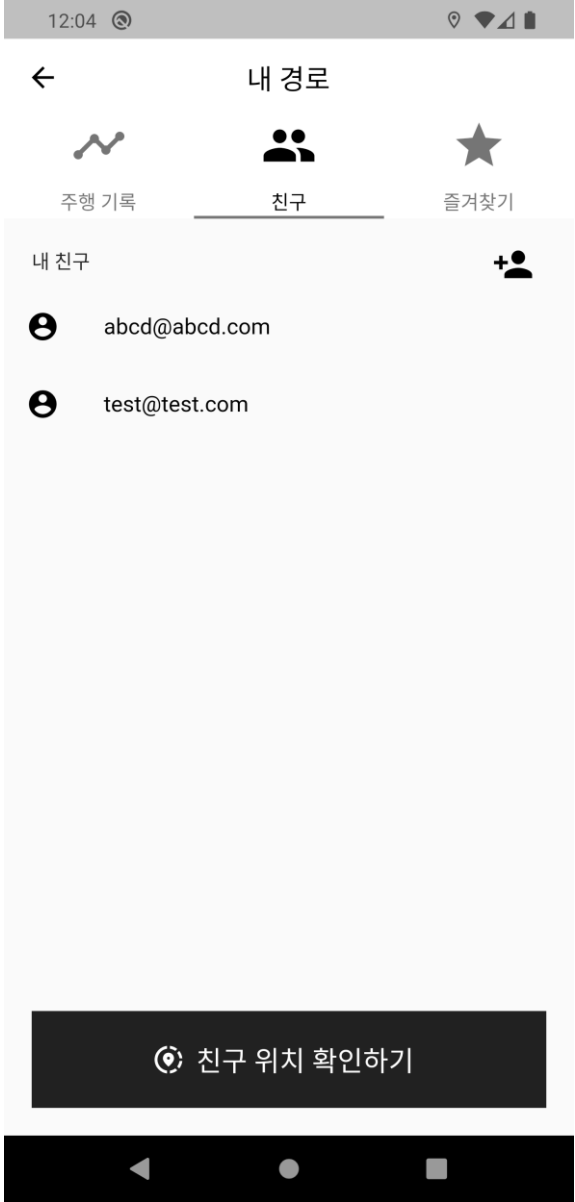
친구 추가



```
Future addfriends(String uid, String name, LatLng latlng) async {  
  await showDialog(  
    context: context,  
    builder: (context) => new AlertDialog(  
      title: Text("친구 추가하시겠습니까?"),  
      actions: <Widget>[  
        new FlatButton(  
          onPressed: () async{  
            await _firestore.collection('users').doc(currUser()).collection('friends').doc(uid).set({  
              'location' : GeoPoint(latlng.latitude, latlng.longitude),  
              'name' : name,  
              'uid' : uid  
            });  
            await _firestore.collection('users').doc(currUser()).collection('friends').doc(currUser()).  
              Navigator.of(context).pop();  
          },  
          child: Text("YES")  
        ), // FlatButton  
        new FlatButton(  
          child: Text("NO"),  
          onPressed: () => Navigator.of(context).pop(false),  
        ), // FlatButton  
      ], // <Widget>[]  
    ), // AlertDialog  
  );  
}
```

user 개인의 friends 문서에
친구 추가할 사용자의 위치 정보, 이메일, uid 필드를 추가

친구 추가

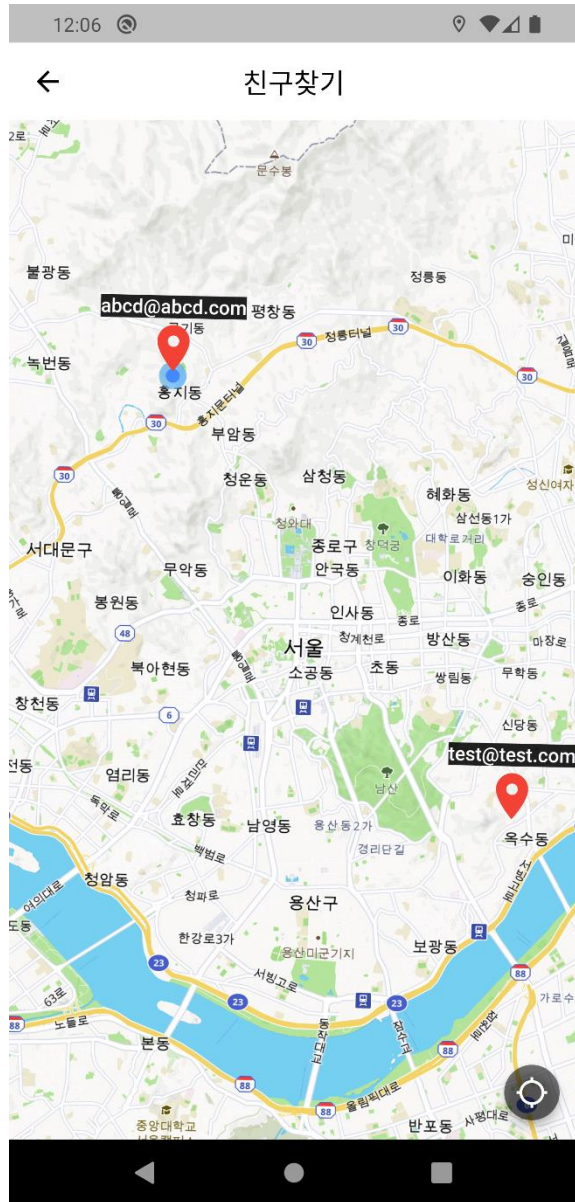


🏠 > users > TcIKbjd0hwao3pMXvVn... > friends > P3LeS3rKJKVngak2iUZdz8wjXGz1		
TcIKbjd0hwao3pMXvVn...	friends	P3LeS3rKJKVngak2iUZdz8wjXGz1
+ 컬렉션 시작	+ 문서 추가	+ 컬렉션 시작
bookmarks	P3LeS3rKJKVngak2iUZdz8wjXGz1	+ 필드 추가
friends >	eKFVcYdTw3PB4lfzC4HawB3	location: [37.6026367° N, 126.9552517° E]
records		name: "abcd@abcd.com"
		uid: "P3LeS3rKJKVngak2iUZdz8wjXGz1"
+ 필드 추가		
location: [37.6026367° N, 126.9552517° E]		
name: "1234@1234.com"		
public: true		
uid: "TcIKbjd0hwao3pMXvVn"		

현재 USER의 friends 컬렉션에 친구 추가한 사용자들 각각의 데이터 저장

현재 USER의 데이터 필드

친구 위치



```
return StreamBuilder<QuerySnapshot>(
  stream: _firestore.collection('users').doc(currUser()).collection('friends').snapshots(),
  builder: (context, snapshot) {
    if (!snapshot.hasData) {
      return Center(
        child: CircularProgressIndicator(),
      ); // Center
    }
    friends.clear();
    locations.clear();
    id.clear();

    final users = snapshot.data!.docs;
    for (var user in users) {
      String uid = user['uid'];
      friends.add(uid);
    }

    for(int i=0; i<uids.length; i++){
      for(int j=0; j<friends.length; j++){
        if(uids[i] == friends[j]) {
          locations.add(locs[i]);
          id.add(names[i]);
        }
      }
    }
  }
)
```

데이터베이스에 마지막으로 저장된
친구의 위치를 지도에 표시

user 실시간 위치

2 Answers

Active Oldest Votes



1



If you are looking to update your Firestore document at a specific time frequency, you may consider [scheduled Cloud Functions](#). This method of updating documents creates a Google Cloud Pub/Sub topic and Cloud Scheduler to trigger events on that topic, which ensures that your function runs on the desired schedule.

Share Improve this answer Follow



Add a comment



0



You can use Timer for update data after specific time.

```
Timer _timer;
int _start = 10;

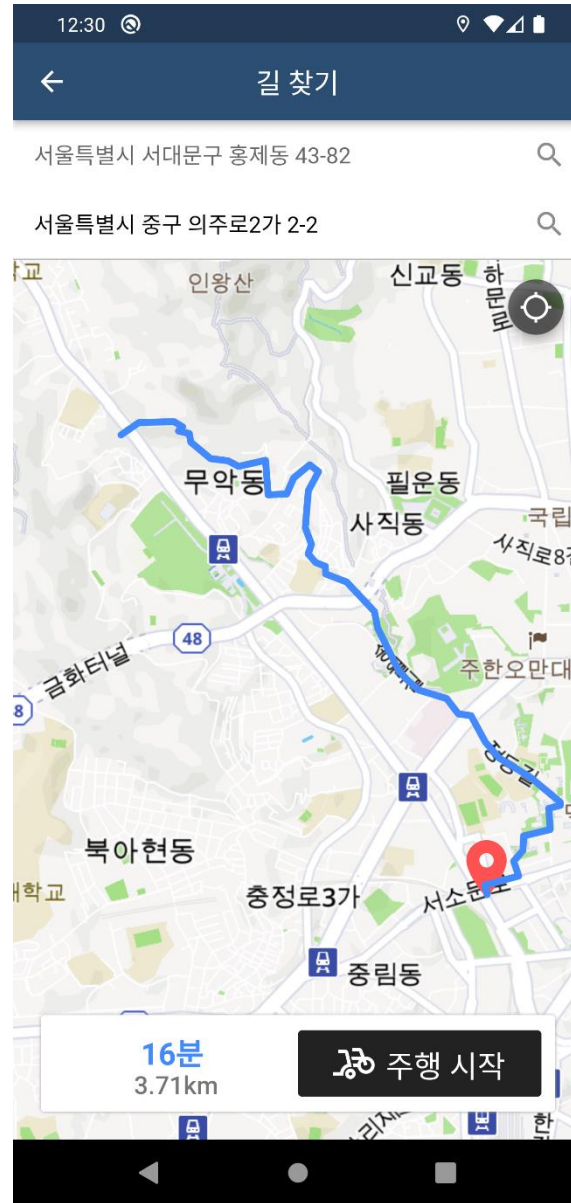
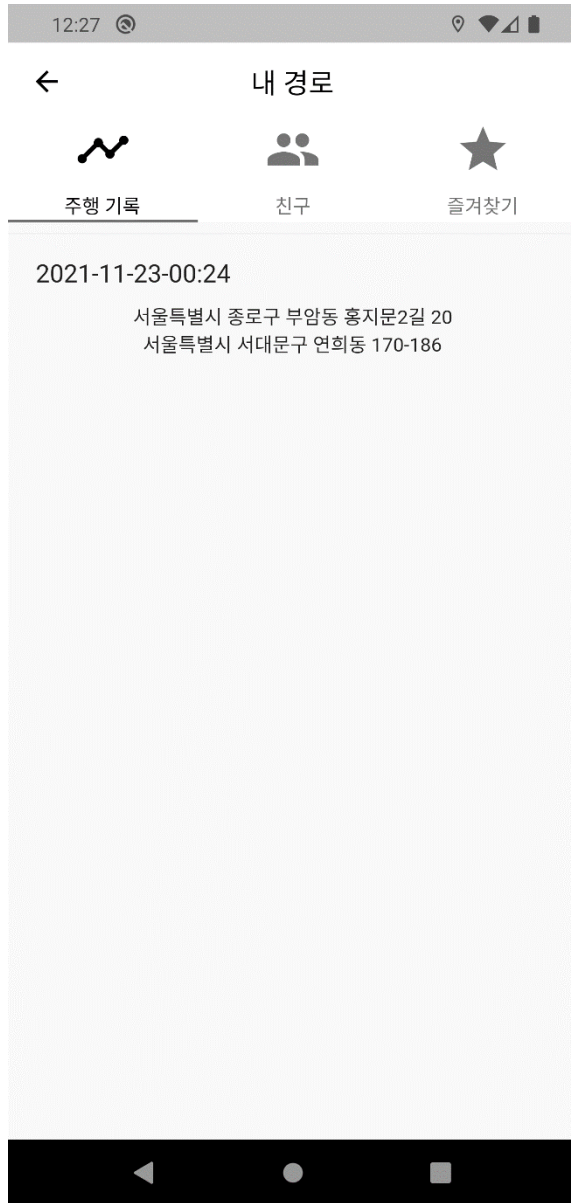
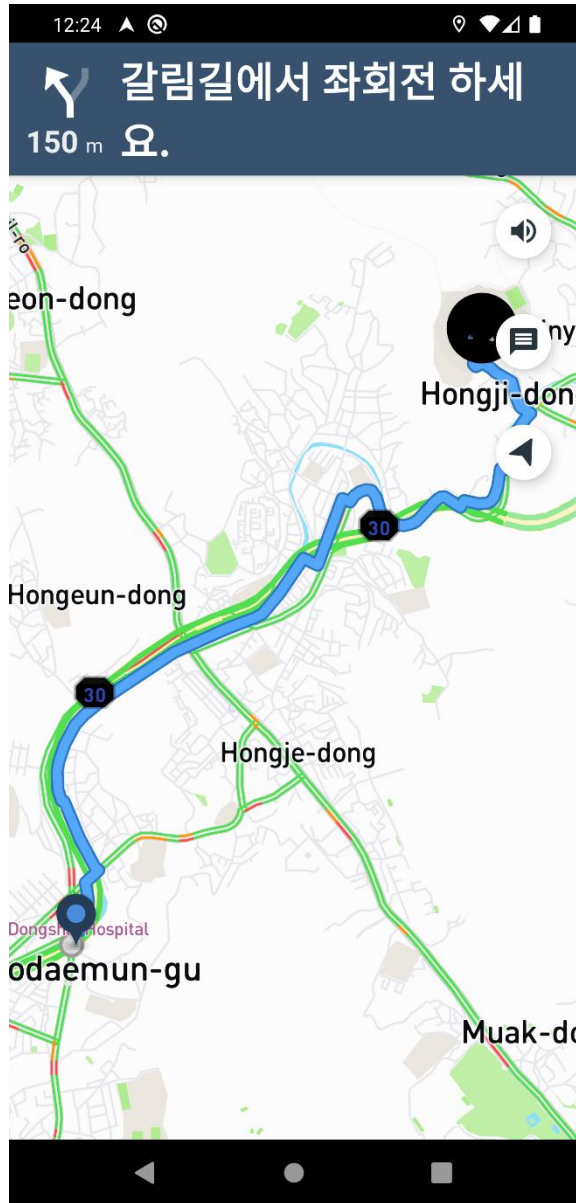
void startTimer() {
  const oneSec = const Duration(seconds: 1);
  _timer = new Timer.periodic(
    oneSec,
    (Timer timer) => setState(
      () {
        if (your logic) {
          //Your code
        } else {
          //Your code
        }
      },
    ),
  );
}
```

특정 시간마다 사용자 위치를 데이터베이스에 update 또는
실시간으로 사용자 위치를 데이터베이스에 update
-> 보완, 테스트 필요

```
Future<void> _listenLocation() async {
  _location = location.onLocationChanged.handleError((onError) {
    print(onError);
    _location?.cancel();
    setState(() {
      _location = null;
    });
  }).listen((loc.LocationData currentlocation) async {
    await _firestore.collection('users').doc(currUser()).update({
      'location': GeoPoint(currentlocation.latitude!, currentlocation.longitude!),
    });
  });
}

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

주행 기록



THANK YOU