RunAnywhere API Manual

❖Database

➢DDL:

```
create table profile (
        uid                 varchar(15)     not null,
        firstName           varchar(64)     not null,
        lastName            varchar(64)     not null,
        profileImg          varchar(200),
        birthdate           DATE            not null,
        phone               varchar(15),
        email               varchar(64),
        gender              char,
        records             dictionary,
        address1            varchar(200),
        address2            varchar(200),
        city                varchar(64),
        state               varchar(15),
        zipcode             varchar(15),
        organization        varchar(64),
        club                varchar(64),
        emergName           varchar(64),
        emergPhone          varchar(15),
        primary key (uid)
) ;

create table race (
        raceId                      int             not null,
        raceName            varchar(32)     not null,
        raceDescription     varchar(800),
        imagePath           varchar(200),
        eventday            DATE            not null,
        regopendate         DATE            not null,
          regclosedate        DATE,
          distance            decimal(5,1)        not null,
          capacity            int,
          startdttm           TIME             not null,
          startLatLng         varchar(32) ,
          gpxFile             varchar(200),
          isOpen              boolean,

          uid                 varchar(15)        not null,
          primary key (raceId),
          unique (name),
          foreign key (uid) references profile(uid)
) ;

create table participate (
               raceId                         varchar(15)        not null,
               uid                   varchar(15)          not null,
               tshirtsize            varchar(15),
               bibnumber             varchar(15),
               chipnumber            varchar(15),
               finishedtime          TIME,
               finisheddistance      varchar(15),
   primary key (raceId, uid),
   foreign key (uid) references member(uid),
   foreign key (raceId) references race(raceId)
) ;


create table  follow(
      followId                varchar(15)          not null,
      Uid                     varchar(15)          not null,
   primary key (followid, uid),
   foreign key (uid) references member(uid),
   foreign key (followid) references member(uid)
) ;
```

```
create table invite (
        raceId                  varchar(15)         not null,
        uid                 varchar(15)         not null,
        followId                varchar(15)         not null,
        isAccept            boolean,            not null,
primary key (raceId,uid,followId),
foreign key (uid) references member(uid),
foreign key (followId) references member(uid),
foreign key (raceId) references race(raceId)
) ;
```

❖Authentication:

1. **redirectToLogIn**()

2. **signOut**()

3. **getCurrentUser**(auth)

❖RealTime Database

Races:

1. **addUpdateRace**(raceId, raceName, raceDescription, imagePath, eventday, startdttm, capacity, distance, startLatLng, gpxFile , isOpen)

2. **deleteRace**(raceId)

3. **getRaces**(cb)
   e.g.
```
        getRaces( function(childSnapshot) {
                // your control code
                //addItem( childSnapshot.key ,childSnapshot.val() );
           });
```

4. **getMyRegisterRaces**(cb)

5. **getMyCreateRaces**(cb)

6. **getRaceRegisters**(raceId, cb)

7. **getRace**(raceID, cb)

8. **getRaceByName**(raceName, cb)
   e.g.
   ```
       getRaceByName(raceName, childSnapshot => {
            // your control code
           //childSnapshot.key ,childSnapshot.val()
         });
   ```

9. **getInviteRaces**(cb)
   e.g.
   ```
   getInviteRaces( function(snapRace, snapProfile) {
       var profileData = snapProfile.val();
       var followName = profileData.firstName + ' ' + profileData.lastName ;
       var raceData = snapRace.val();
   ```

Profile:

1. **addUpdateProfile**(profileImg,
   firstName, lastName, email, gender, birthdate, phone, address1, address2, city, state,zipcode)

2. **getMyProfil**e(cb)
   E.g.
   ```
   getProfile(snap =>  console.log(snap.val()) );
   ```

3. **getProfile(profileId, cb)**

4. **getStatInfo**(peopleId, cb)
   E.g.

```
getStatInfo( "-MMT6shfN4bTHRQY42p9",
function( count, distance, time, records) {
        console.log( count );
        if(records["Marathon"] != null)
            //Half Marathon, 5K, 100Miles, 50Miles
        {
          console.log(records["Marathon"] );
        }
      } );
```

5. **getMyStat**(cb)
   E.g.
   getProfile(snap => console.log(snap.val()));

Participate:

1. **addRegister**(raceId)

2. **cancelRegister**(raceId)

3. **updatePanticipateInfo**(raceId, panticipateid, finishedtime, finisheddistance)

4. **updateMyResultInfo**(raceId,finishedtime, finisheddistance)

5. **getRegisterRaces**(peopleId, cb)
   E.g.
   getRegisterRaces("dKNyWQ5BfDUhw91Ru0tDJ36ir4I2", snap => console.log(snap.val()));

6. **getRaceResults**(raceId, cb)
   E.g.
   getRaceResults("-MJxxESa1SpS-6mHlv6N", (resultsnap, profilesnap) => {

```
        console.log(profilesnap);
        console.log(profilesnap.val().firstName + " " +
        profilesnap.val().lastName);
        console.log(resultsnap.val().finishedtime);

      });
```

7. **getPeopleResults**(peopleId, cb)
   E.g.
   getPeopleResults("dKNyWQ5BfDUhw91Ru0tDJ36ir4I2", (resultsnap, racesnap) => {
   ```
           console.log(racesnap.val().raceName );
           console.log(resultsnap.val().finishedtime);

         });
   ```

8. **getMyRacesByEventday(cb)**

9. **getRegisterRacesByEventDay(peopleId, cb)**

10. **isMyRegisterRace(raceId,cb)**

Follow:

1. **addFollow**(peopleId)

2. **Unfollow**(peopleId)

3. **getFollows**(cb)
   E.g.
   getFollows(snap => console.log(snap.val()));

4. **getUninvitedFollows(raceId, cb)**
   E.g.
   getFollows(snap => console.log(snap.val()));

5. **getFollowsActivities**(cb)
   e.g.
   getFollowsActivities(function(childSnapshotRace, snapProfile, childSnapshotParticipate){
   ```
    var profileData = snapProfile.val();
    var followName = profileData.firstName + ' ' + profileData.lastName ;
   var resultData = childSnapshotParticipate.val();
   var raceData = childSnapshotRace.val();
   ```

Invites:

1. **inviteRace**(raceId, peopleId)

2. **removeInvite**(peopleId)

3. **isInvited(raceId, followId, cb)**

Search:

1. **searchRaceByName**(raceName, cb)
   E.g.
   searchRaceByName("Ke",snap => console.log(snap.val().raceName));

2. **searchRaceByFilter**(raceDistanceMin, raceDistanceMax, raceMonthMin, raceMonthMax,cb)
   E.g.
   searchRaceByFilter(50, 100, 1,12,snap => console.log(snap.val()));

3. **searchRaceByDistanceDate**(raceDistanceMin, raceDistanceMax, raceDateFrom, raceDateTo,cb)

4. **searchPeopleByGenderAge**(gender, ageMin, ageMax, cb)

5. **searchPeopleByName**(peopleName, cb)

E.g.
```
searchPeopleByName("James",snap => console.log(snap.val()));
```

6. **searchPeopleByFilter**(peopleGender, peopleBirthMin, peopleBirthMax, cb)
   E.g.
   ```
   searchPeopleByFilter("F", "1988-1-1", "2004-1-1",snap =>
   console.log(snap.val()));
   ```

❖Files Storage:

1. **uploadProfilePicture**(raceName, cb)
   E.g.
   ```
   var fileUpload = document.getElementById("cameraInput");
    fileUpload.addEventListener('change', function(evt) {
         var firstFile = evt.target.files[0]; // upload the first file only
         console.log(firstFile);
         uploadProfilePicture("test", firstFile, url =>{
           console.log(url);
           var img = document.getElementById("raceimg");
           img.setAttribute( "src", url);
          //also update to firebase
         });
     });
   ```

2. **uploadRaceLogo**(raceDistanceMin, raceDistanceMax, raceMonthMin, raceMonthMax,cb)

3. **uploadRaceKML**(peopleName, cb)

4. **uploadRacePictures**(peopleName, cb)

❖Map:

1. **initMap**()
2. **initMapForId**(controlId, kmlFile)
3. **getClickLatLng**()
4. **setRouteWithMap**(map, kmlPath)
5. **setRoute**(kmlPath)
6. **setRacesMarker**(raceId, raceData)
7. **setMapCenter**(startLatLng)
8. **markCurrentLocation**()
9. **searchOnMap**()