



**isReferee**  
Android App

JANUARY 9, 2021

AIN SHAMS UNIVERSITY – FACULTY OF ENGINEERING

**AIN-SHAMS UNIVERSITY**

**FACULTY OF ENGINEERING**

**CSE 430: ADVANCED MOBILE COMPUTING**



## **isReferee**

**Name: Abdelrahman Amr El-Sayed Mohamed Issawi**

**ID: 16P6001**

**Email: aid-issawi@hotmail.com**

Submitted to:

**Dr. Haytham Azmi.**

**Eng. Fatema Mahmoud**

**JANUARY 9, 2021**

A report for Selected Topics Course coded CSE430 with the requirements of

Ain Shams University.

## Table of Contents

INTRODUCTION.....	4
Old Scenario Disadvantages.....	5
Proposed Solution.....	5
DATABASE .....	5
HARDWARE .....	6
Circuit.....	6
Code .....	6
VIEWS DESIGN .....	8
FEATURES.....	9
TESTING SCENARIOS .....	9
Normal Sequence of Application .....	9
Odd Trials.....	9
FIREBASE .....	10
Firebase Authentication.....	10
Cloud Firestore.....	10
Realtime Database .....	10
CODES SAMPLES .....	11
Model.....	11
Dao .....	11
Repository Sample .....	11
Room Database Sample .....	11
View .....	12
Main Activity Sample .....	12
Content Records Sample.....	12
Adapter Sample .....	12
View Model.....	13
View Model Sample .....	13
XML.....	13
Custom Styles Sample.....	13
Custom Toolbar.....	13
GIT.....	14
DEMO VIDEO.....	14

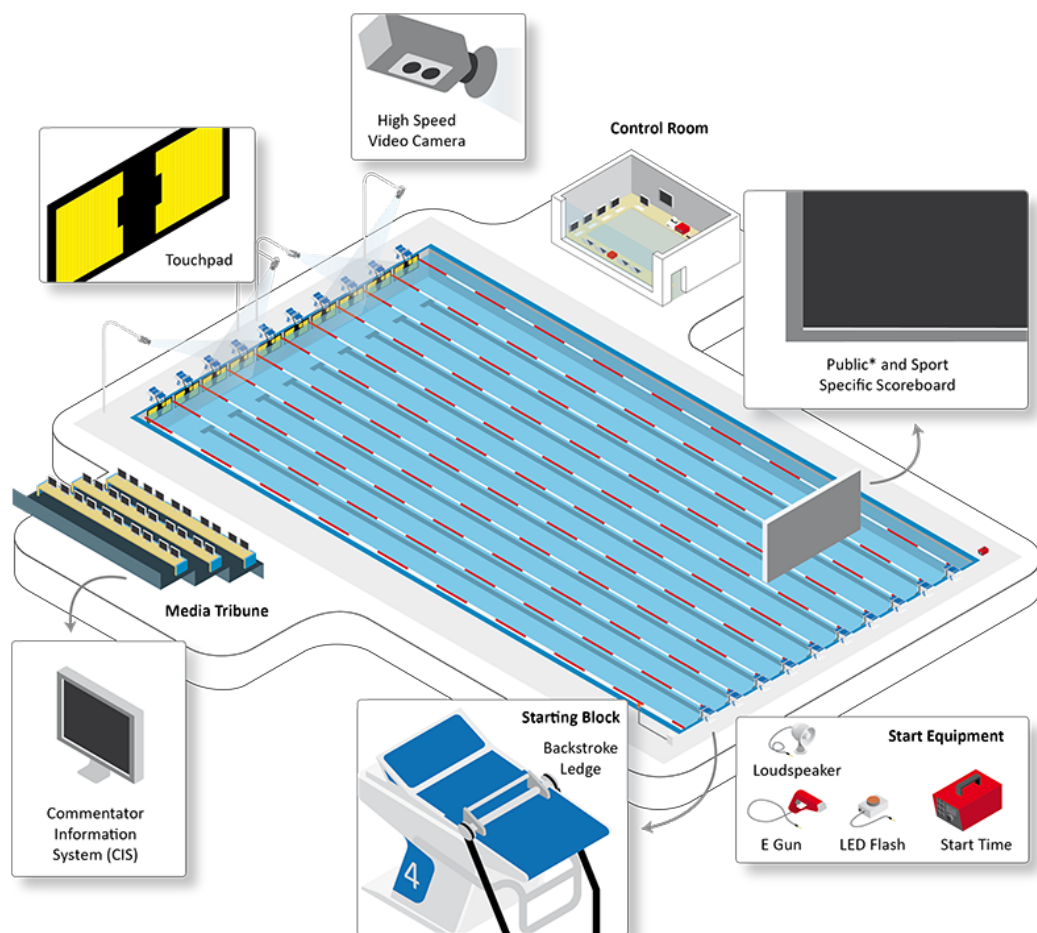
## INTRODUCTION

I am interested in improving my sport, which is Modern Pentathlon. In this application, I am focusing on presenting a solution for the swimming event. The goal is to automate the process and minimize human interaction and redundant work which is too much.

I'll show a little demo to clarify my concept and the old scenario before automating and improving it.

Let's imagine having 4 lanes swimming pool and 40 players in the championship. Then we are going to distribute them to 10 heats "Groups" each group contains 4 players. And now starting with heat number 1. The first 4 players start swimming when the referee says "Heat Number 1, Take your mark, start" and this referee uses the start equipment in the below figure.

The lane referee starts the stopwatch time and the global stopwatch displayed to the public is started "Scoreboard", when the swimmers arrive the lane referee stops the stopwatch time and whenever a swimmer touches the lane **touchpad** the record is automatically printed to a paper and the referee write his time on another paper then one referee compares both records, and if matched another referee write these records in the web application or the database.



### Old Scenario Disadvantages

The main issue is the redundant work both the lane referee and the system produce times and the comparison process and then the writing process. This is time-consuming. Also, many humans are involved.

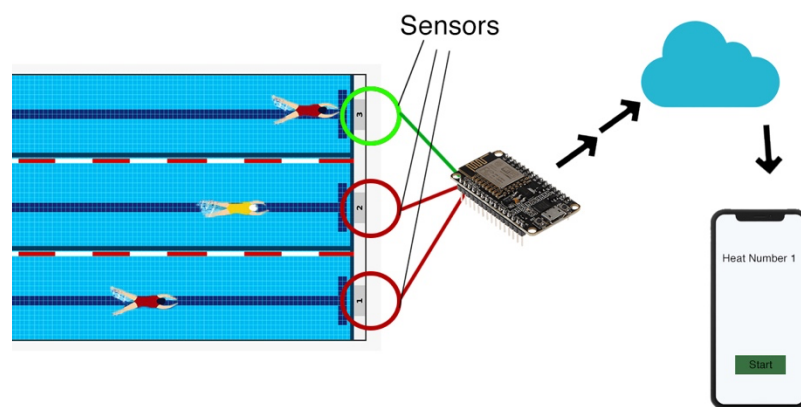
### Proposed Solution

My proposed solution is to overcome this and directly save time to the database. Decreasing the cost, human interaction, and the time.

And this would happen either by updating the infrastructure to navigate the records to the database instead of printing it, but this is a legacy system and hard to manipulate.

Or, we can provide an internet connection and when the touchpad is touched an interrupt is triggered, so a value associated with the lane in the Realtime database is changed so the app listens to the change and updated the database, and the display scoreboard.

For the second solution, I only need a mobile app that will start the Heat, switch heats, receive records, saves them, and I added an option to retrieve a specific time.



## DATABASE

I used Room Database, Saving the Heat Number, Lane Number, and the time associated with them.

```
@Entity(tableName = "recordsTable", primaryKeys = {"HeatNumber", "LaneNumber"})
public class Record {
    @NonNull @ColumnInfo(name = "HeatNumber")
    public int HeatNum;

    @NonNull @ColumnInfo(name = "LaneNumber")
    public String LaneNum;

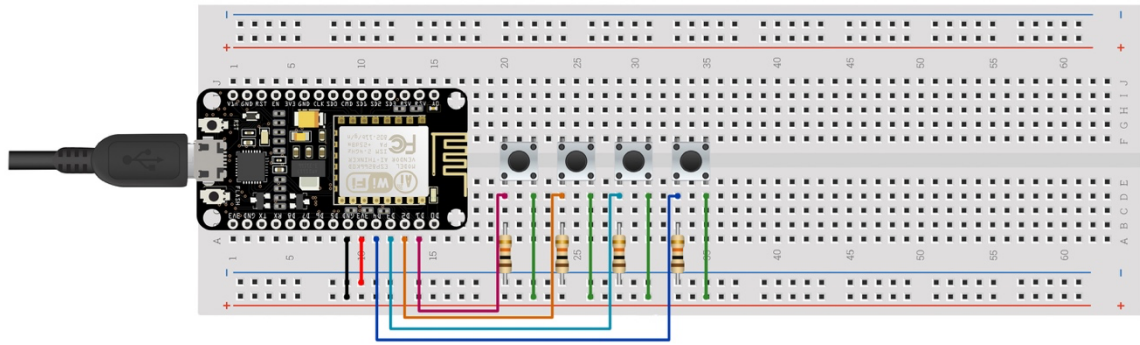
    @NonNull @ColumnInfo(name = "Record")
    public String Record;

    public Record(@NonNull int HeatNum, @NonNull String LaneNum, @NonNull String Record) {
        this.HeatNum = HeatNum;
        this.LaneNum = LaneNum;
        this.Record = Record;
    }

    public int getHeatNumber(){return this.HeatNum;}
    public String getLaneNumber(){return this.LaneNum;}
    public String getRecord(){return this.Record;}
}
```

# HARDWARE

## Circuit



## Code

```
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>

#define WIFI_SSID "doda"
#define WIFI_PASSWORD "@170570r"
#define FIREBASE_HOST "isreferee-5a163-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "NZwshtu141DeKMPzueCdDZu8ef5OPiJpraBXdSmi"

uint8_t GPIO_Pin1 = D1;
uint8_t GPIO_Pin2 = D2;
uint8_t GPIO_Pin3 = D3;
uint8_t GPIO_Pin4 = D4;

int switchOneState = 0;
int switchTwoState = 0;
int switchThreeState = 0;
int switchFourState = 0;

void ICACHE_RAM_ATTR IntCallback1 ();
void ICACHE_RAM_ATTR IntCallback2 ();
void ICACHE_RAM_ATTR IntCallback3 ();
void ICACHE_RAM_ATTR IntCallback4 ();

void setup() {
  Serial.begin(9600);
  pinMode(GPIO_Pin1, INPUT);
  pinMode(GPIO_Pin2, INPUT);
  pinMode(GPIO_Pin3, INPUT);
  pinMode(GPIO_Pin4, INPUT);

  attachInterrupt(digitalPinToInterrupt(GPIO_Pin1), IntCallback1, HIGH);
  attachInterrupt(digitalPinToInterrupt(GPIO_Pin2), IntCallback2, HIGH);
  attachInterrupt(digitalPinToInterrupt(GPIO_Pin3), IntCallback3, HIGH);
  attachInterrupt(digitalPinToInterrupt(GPIO_Pin4), IntCallback4, HIGH);

  // connect to wifi.
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.println();
  Serial.print("connecting to WIFI");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}
```

```

void loop() {
  if(switchOneState == 1) {
    switchOneState = 0;
    Serial.println("switch 1 pressed");
    if(WiFi.status() == WL_CONNECTED) {
      Firebase.setString("lane1", "1");
    }else{
      Serial.println("FAILED");
    }
  }
  if(switchTwoState == 1) {
    switchTwoState = 0;
    Serial.println("switch 2 pressed");
    if(WiFi.status() == WL_CONNECTED){
      Firebase.setString("lane2", "1");
    }else{
      Serial.println("FAILED");
    }
  }
  if(switchThreeState == 1) {
    switchThreeState = 0;
    Serial.println("switch 3 pressed");
    if(WiFi.status() == WL_CONNECTED){
      Firebase.setString("lane3", "1");
    }else{
      Serial.println("FAILED");
    }
  }
  if(switchFourState == 1) {
    switchFourState = 0;
    Serial.println("switch 4 pressed");
    if(WiFi.status() == WL_CONNECTED){
      Firebase.setString("lane4", "1");
    }else{
      Serial.println("FAILED");
    }
  }
  delay(200);
}

void IntCallback1(){
  switchOneState = 1;
}

void IntCallback2(){
  switchTwoState = 1;
}

void IntCallback3(){
  switchThreeState = 1;
}

void IntCallback4(){
  switchFourState = 1;
}

```

# VIEWS DESIGN


IsReferee

Email Address

Password

LOGIN

NOT A REGISTERED REFEREE?, REGISTER NOW!



IsReferee

By Abdelrahman Issawi

Abdelrahman Issawi

aid-issawi@hotmail.com

Records

Get Record

LOGOUT

Heat Number

Lane Number

GET RECORD

00:00:00

Full Name

Email Address

Password

PhoneNumber

☐ Register as Manager ☐ Register as Referee

CREATE ACCOUNT

ALREADY HAVE AN ACCOUNT?, LOGIN NOW.

Choose the Distance

By Abdelrahman Issawi

☒ 50 Meter

☐ 100 Meter

☐ 200 Meter

Heat Number

Lane Number

1

4

GET RECORD

00:13:58

IsReferee

Heat Number 1

Race Distance: 50

00:00:00

Heat#	Lane#	Record
1	1	00:04:36
Heat#	Lane#	Record
1	2	00:07:78
Heat#	Lane#	Record
1	3	00:10:87
Heat#	Lane#	Record
1	4	00:12:08
Heat#	Lane#	Record
2	1	00:04:61
Heat#	Lane#	Record

11:43 PM

Wednesday, January 6

Wi-Fi Mobile data Bluetooth Sound Flashlight

IsReferee • Just now

FINISHED

Heat Number 1 is Finished

00:00:00



## FEATURES

- **Text to Speech** to start the heat.
- **Notifications** to alert the referee when the heat is finished.
- **Async Task** implemented to retrieve records, delete, or insert.
- **Stopwatch** implementation to show the time.
- **Buttons** to start, stop, reset time, and increment or decrement heats.
- Android Architecture **MVVM** to organize the code.
- **Firestore** to save user's data.
- **Firestore Realtime database** to monitor hardware change.
- Users can retrieve any time by lane number and heat number.
- Used **customized** styles, not the default ones, designed the logos, edited the pictures I got.
- **Customized** toolbars.
- Used recycler views, and drawer layouts.
- Used **Live data** to show lanes changing, users, records.
- Used **NodeMCU** for the communication between hardware and firebase.
- Created the circuit demonstrating 4 lanes by **push buttons**.
- Instead of polling, I used **interrupts**.

## TESTING SCENARIOS

### Normal Sequence of Application

Referee Starts the app to register, enter valid information, and then click create an account, or the Referee can login if already registered. Navigates to the home screen where the Referee can start the time change the heat, stop the heat, and reset.

The Referee also after stopping the heat can change the distance, open the records view to see all records or open the get records view to get a specific record.

### Odd Trials

Whenever the Referee clicks the back button or any button except the stop one when the heat is running a toast will show up warning. The heat must be stopped to change the heat number or open the records panel or changing the distance, in the retrieval of specific records the invalid entry numbers are handled as empty fields or not existing heat or lane numbers.

When the button is clicked more than one time. The record is replaced, and this is a valid solution in my use case.

# FIREBASE

## Firebase Authentication

Search by email address, phone number, or user UID					Add user		
Identifier	Providers	Created	Signed In	User UID			
mohanad@gmail.com		Dec 15, 2020	Dec 16, 2020	21MNeWyTXbSN0fcF2CaStgXFXA...			
omarali@gmail.com		Dec 15, 2020	Dec 16, 2020	8huujYzZarW7OKGu31ZHsa4siEf2			
mohamed678140@gmail.com		Dec 31, 2020	Dec 31, 2020	IUat77F9wXg9VFDDZDpnq4BDHa03			
zaid@gmail.com		Dec 19, 2020	Dec 19, 2020	SaBB871tBIX5Rp38dTavRDVqqQI2			
aid-issawi@hotmail.com		Dec 15, 2020	Jan 5, 2021	SbwmiThIQWVX8sCrleKZ4FFF0b...			
nada@gmail.com		Dec 18, 2020	Dec 18, 2020	bTfxXYj2Aih2DSPiAzsFzcnEo53			
aid.issawi@hotmail.com		Dec 15, 2020	Dec 15, 2020	eSjV23MQcmQ3zTeulqoY1LJeTtq1			
					Rows per page:	50	1 - 7 of 7

## Cloud Firestore

isreferee-5a163

Users

+ Start collection

Users

+ Add document

21MNeWyTXbSN0fcF2CaStgXFXA02

6iHSAKyNRRWgChFmoudkbQbkvRG2

DPxfGhuVUsRS9WxpM5ysVDe1Sh2

IUat77F9wXg9VFDDZDpnq4BDHa03

SaBB871tBIX5Rp38dTavRDVqqQI2

SbwmiThIQWVX8sCrIeKZ4FFF0bm2

bTfxXYj2Aih2DSPiAzsFzcnEo53

eSjV23MQcmQ3zTeulqoY1LJeTtq1

frFf2BMjAuXu99qZw9zp1P78ECM2

SbwmiThIQWVX8sCrIeKZ4FFF0bm2

+ Start collection

+ Add field

FullName: "Abdelrahman Issawi"

PhoneNumber: "01093511759"

UserEmail: "aid-issawi@hotmail.com"

isAdmin: "1"

## Realtime Database

https://isreferee-5a163-default-rtdb.firebaseio.com/

isreferee-5a163-default-rtdb

distance: "50"

lane1: "0"

lane2: "0"

lane3: "0"

lane4: "0"

started: "0"

Database location: United States (us-central1)

## CODES SAMPLES

### Model

#### Dao

```
@Dao
public interface RecordDao {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insert(Record record);

    @Query("DELETE FROM recordsTable")
    void deleteAll();

    @Delete()
    void delete(Record record);

    @Query("SELECT * from recordsTable ORDER BY HeatNumber, LaneNumber ASC")
    LiveData<List<Record>> getAllRecords();

    @Query("SELECT Record from recordsTable where HeatNumber = :number AND LaneNumber = :lane")
    String getRecord(int number, int lane);
}
```

#### Repository Sample

```
public Repository(Application application) {
    RecordsRoomDatabase db = RecordsRoomDatabase.getDatabase(application);
    RecordDao = db.RecordDao();
    AllRecords = RecordDao.getAllRecords();
    this.application = application;
    userMutableLiveData = new MutableLiveData<>();

    lane1Data = new MutableLiveData<>();
    lane2Data = new MutableLiveData<>();
    lane3Data = new MutableLiveData<>();
    lane4Data = new MutableLiveData<>();
    startedData = new MutableLiveData<>();
    fAuth = FirebaseAuth.getInstance();
    fStore = FirebaseFirestore.getInstance();

    if(FirebaseAuth.getInstance().getCurrentUser() != null){
        userMutableLiveData.postValue(fAuth.getCurrentUser());}

    int theatNumber = 0;
    String tlaneNumber = "0";
    String ttime = "00:00:00";
    Record trecord = new Record(theatNumber, tlaneNumber, ttime);
    insert(trecord);
    delete(trecord);
}
```

#### Room Database Sample

```
public static RecordsRoomDatabase getDatabase(final Context context) {
    if (INSTANCE == null) {
        synchronized (RecordsRoomDatabase.class) {
            if (INSTANCE == null) {
                INSTANCE = Room.databaseBuilder(context.getApplicationContext(),
                    RecordsRoomDatabase.class, "Records_database")
                    // Wipes and rebuilds instead of migrating
                    // if no Migration object.
                    // Migration is not part of this practical.
                    .fallbackToDestructiveMigration()
                    .addCallback(sRoomDatabaseCallback)
                    .build();
            }
        }
    }
    return INSTANCE;
}
```

## View

### Main Activity Sample

```
MainViewModel.getLane1Changing().observe(this, new Observer<String>() {
    @Override
    public void onChanged(String laneChange1) {

        if(laneChange1.equals("1") && isStarted)
        {
            lane1 = timer.getText().toString();
            Log.d("LANE1", "----- " + lane1);
            int heatNumber = count;
            String laneNumber = "1";
            String time = lane1;
            Record record = new Record(heatNumber, laneNumber, time);
            MainViewModel.insert(record);
            MainViewModel.setLane1();
        }
        else if(laneChange1.equals("1") && !isStarted)
        {
            MainViewModel.setLane1();
        }
    }
});
```

### Content Records Sample

```
RecordsViewModel = ViewModelProviders.of(this).get(ViewModel.class);
RecordsViewModel.getAllRecords().observe(this, new
Observer<List<Record>>() {
    @Override
    public void onChanged(@Nullable final List<Record> records) {
        // Update the cached copy of the words in the adapter.
        adapter1.setRecords(records);
    }
});
}
```

### Adapter Sample

```
@Override
public void onBindViewHolder(RecordViewHolder holder, int position) {
    //if ((Integer) position ==null ) position = 1;
    Log.d("TESTMVVM", "POSITION Record " + position);
    if (Records != null) {

        Record current = Records.get(position);
        int heat = current.getHeatNumber();

        holder.HeatNumber.setText(Integer.toString(current.getHeatNumber()));
        holder.LaneNumber.setText(current.getLaneNumber());
        holder.Record.setText(current.getRecord());

    } else {
        // Covers the case of data not being ready yet.
        holder.Record.setText("00");
    }
}
```

## View Model

### View Model Sample

```
public ViewModel(Application application) {
    super(application);
    mRepository = new Repository(application);
    userMutableLiveData = mRepository.getUserMutableLiveData();

    lane1Data = mRepository.getLane1Changing();
    lane2Data = mRepository.getLane2Changing();
    lane3Data = mRepository.getLane3Changing();
    lane4Data = mRepository.getLane4Changing();
    startedData = mRepository.getStartedRef();

    AllRecords = mRepository.getAllRecords();
}
```

## XML

### Custom Styles Sample

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" android:padding="30dp">
    <solid android:color="#FFFFFF" />
    <stroke
        android:width="2dp"
        android:color="#3D3B3C" />
    <corners
        android:bottomRightRadius="10dp"
        android:bottomLeftRadius="10dp"
        android:topLeftRadius="10dp"
        android:topRightRadius="10dp" />
</shape>
```

### Custom Toolbar

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="12dp"
    android:gravity="center_vertical"
    >
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="ClickMenu"
        android:src="@drawable/menuicon"/>

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@string/app_name"
        android:textSize="24sp"
        android:textStyle="bold"
        android:textAlignment="center"
        android:textColor="#3D3B3C" />

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="ClickDistance"
        android:src="@drawable/moreicon"
        />
    <!-- android:onClick="ClickDistance"-->
    <!-- android:onClick="DistanceMenu"-->
</LinearLayout>
```

## GIT

<https://github.com/Issawi98/isReferee-Android-App>

## DEMO VIDEO

[https://drive.google.com/file/d/1qR2-pANDOT6m8TsYfKHccZdrUytdLT\\_m/view?usp=sharing](https://drive.google.com/file/d/1qR2-pANDOT6m8TsYfKHccZdrUytdLT_m/view?usp=sharing)