Mobile Application Development

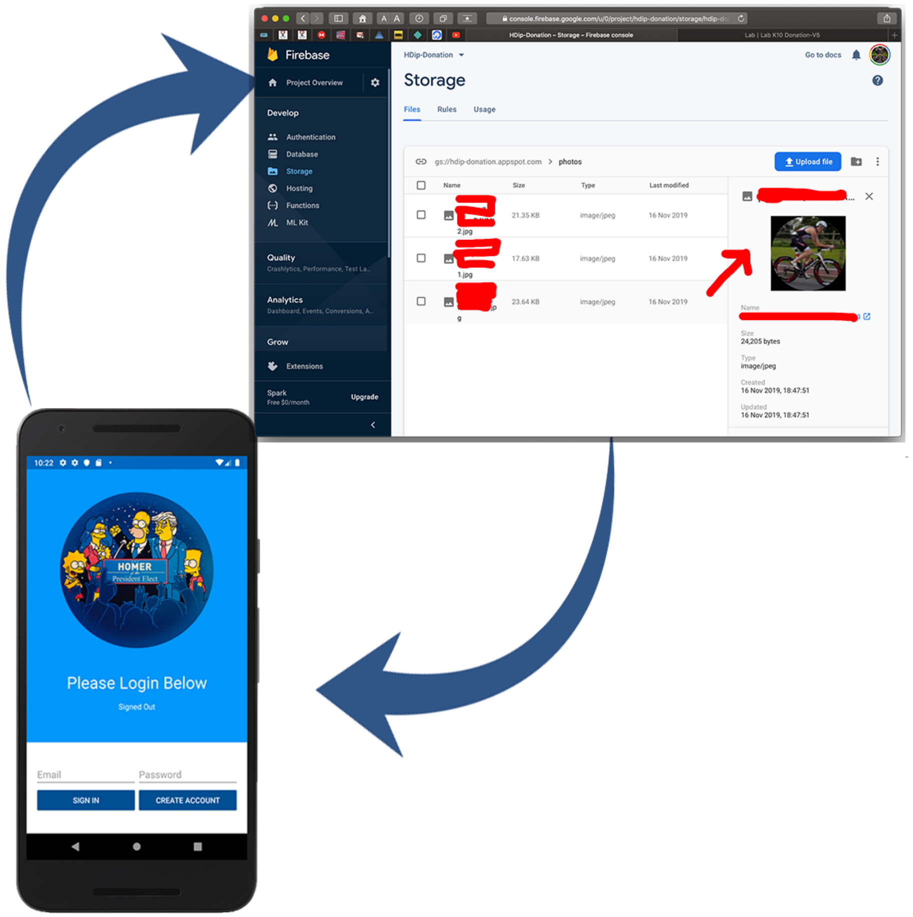
Produced by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology
http://www.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

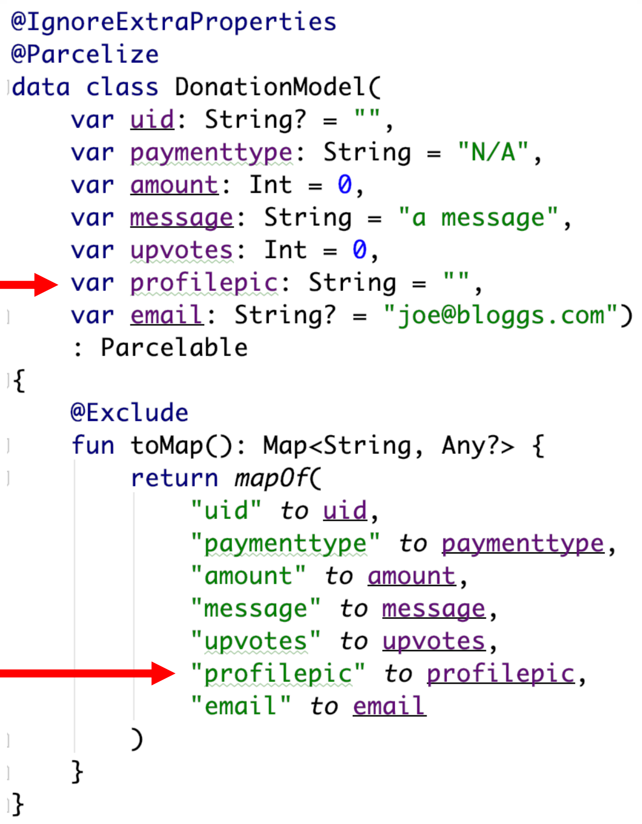
Donation-V6 Walkthrough



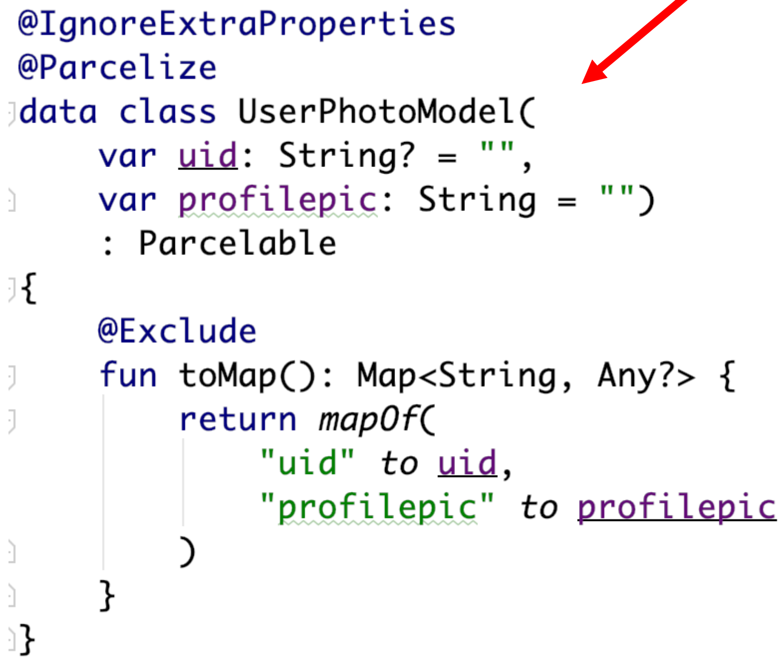
https://github.com/firebase/quickstart-android

Refactor Models

New **UserPhotoModel** class



```kotlin
@IgnoreExtraProperties
@Parcelize
data class DonationModel(
    var uid: String? = "",
    var paymenttype: String = "N/A",
    var amount: Int = 0,
    var message: String = "a message",
    var upvotes: Int = 0,
    var profilepic: String = "",
    var email: String? = "joe@bloggs.com")
    : Parcelable
{
    @Exclude
    fun toMap(): Map<String, Any?> {
        return mapOf(
            "uid" to uid,
            "paymenttype" to paymenttype,
            "amount" to amount,
            "message" to message,
            "upvotes" to upvotes,
            "profilepic" to profilepic,
            "email" to email
        )
    }
}
```



```kotlin
@IgnoreExtraProperties
@Parcelize
data class UserPhotoModel(
    var uid: String? = "",
    var profilepic: String = "")
    : Parcelable
{
    @Exclude
    fun toMap(): Map<String, Any?> {
        return mapOf(
            "uid" to uid,
            "profilepic" to profilepic
        )
    }
}
```

Introduce Storage Reference

```
class DonationApp : Application(), AnkoLogger {

    lateinit var auth: FirebaseAuth
    lateinit var database: DatabaseReference
    lateinit var googleSignInClient: GoogleSignInClient
    lateinit var storage: StorageReference
    lateinit var userImage: Uri

    override fun onCreate() {
        super.onCreate()
        info("Donation App started")
    }
}
```

Our **Storage** instance

Current Users Image ref

Introduce Storage Reference (UpdateUI( ) in Login)

```
app.database = FirebaseDatabase.getInstance().reference
app.storage = FirebaseStorage.getInstance().reference

startActivity<Home>()
```

Our **Storage** instance

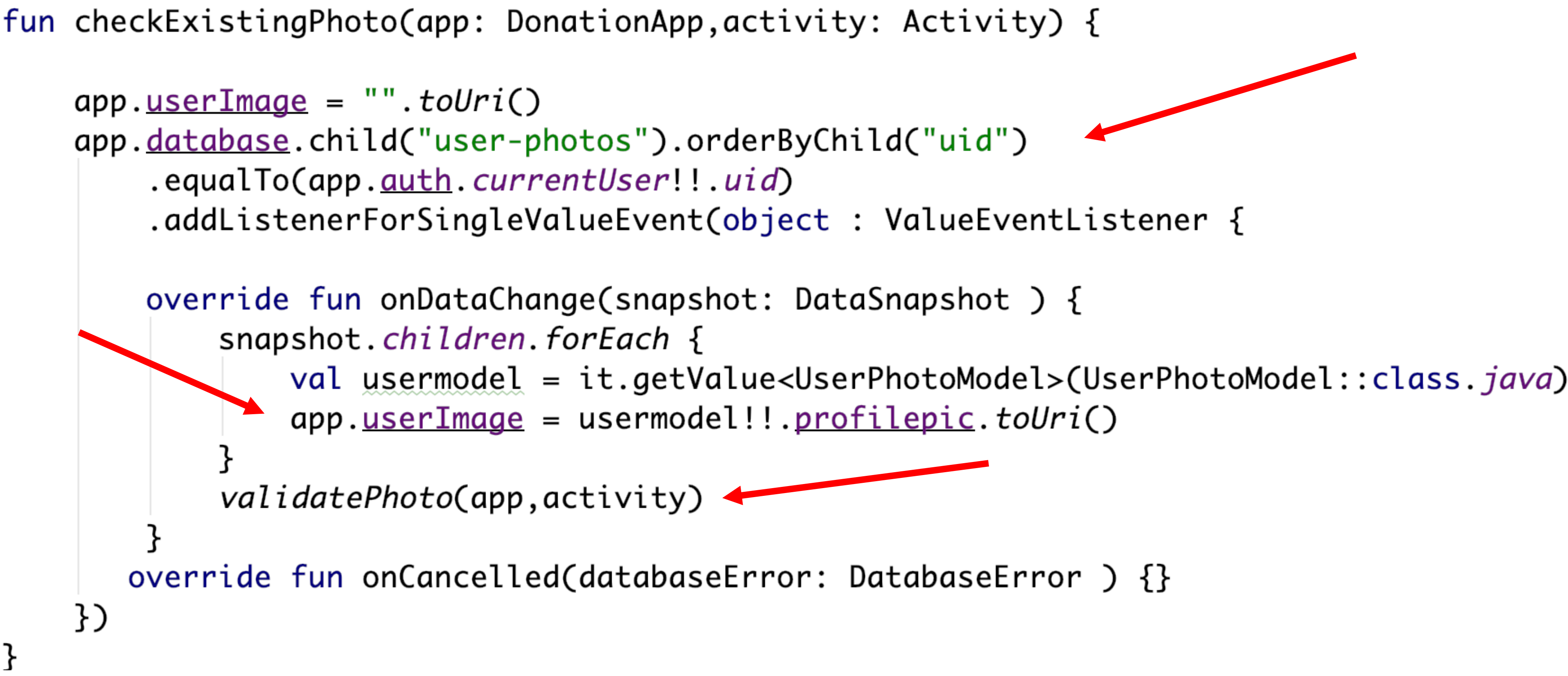
Introduce Photo Validation Process

```
checkExistingPhoto(app,this)
    validatePhoto(app,activity)
        uploadImageView(app,
            activity.navView.getHeaderView(0).imageView)
                updateAllDonations(app)
                    writeImageRef(app,app.userImage.toString())
```

Updates
storage

Updates
database

Introduce Photo Validation Process



```
fun checkExistingPhoto(app: DonationApp,activity: Activity) {

    app.userImage = "".toUri()
    app.database.child("user-photos").orderByChild("uid")
        .equalTo(app.auth.currentUser!!.uid)
        .addListenerForSingleValueEvent(object : ValueEventListener {

        override fun onDataChange(snapshot: DataSnapshot ) {
            snapshot.children.forEach {
                val usermodel = it.getValue<UserPhotoModel>(UserPhotoModel::class.java)
                app.userImage = usermodel!!.profilepic.toUri()
            }
            validatePhoto(app,activity)
        }
       override fun onCancelled(databaseError: DatabaseError ) {}
    })
}
```

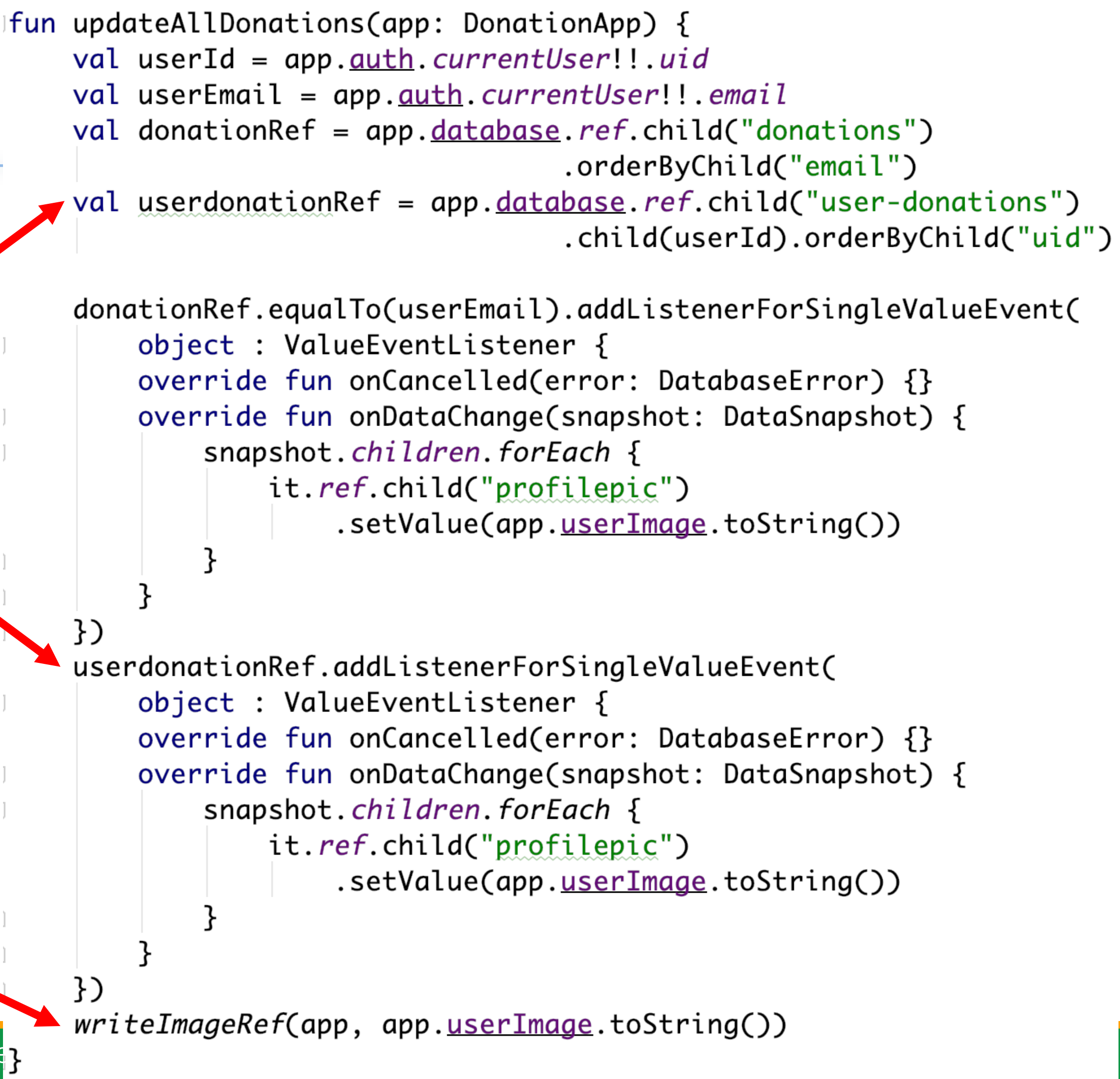
Introduce Photo Validation Process

```kotlin
fun validatePhoto(app: DonationApp, activity: Activity) {
    var imageUri: Uri? = null
    val imageExists = app.userImage.toString().length > 0
    val googlePhotoExists = app.auth.currentUser?.photoUrl != null

    if(imageExists) imageUri = app.userImage
    else if (googlePhotoExists) imageUri = app.auth.currentUser?.photoUrl!!

    if (googlePhotoExists || imageExists) {
        if(!app.auth.currentUser?.displayName.isNullOrEmpty())
        activity.navView.getHeaderView(0)
            .nav_header_name.text = app.auth.currentUser?.displayName
        else
            activity.navView.getHeaderView(0)
                .nav_header_name.text = "Donation (Lab K11)"

        Picasso.get().load(imageUri)
            .resize(180, 180)
            .transform(CropCircleTransformation())
            .into(activity.navView.getHeaderView(0).imageView, object : Callback {
                override fun onSuccess() {
                    uploadImageView(app,
                        activity.navView.getHeaderView(0).imageView)
                }
                override fun onError(e: Exception) {}
            })
    }
    else {   // New Regular User, upload default pic of homer
        activity.navView.getHeaderView(0).imageView
            .setImageResource(R.mipmap.ic_launcher_homer_round)
        uploadImageView(app, activity.navView.getHeaderView(0).imageView)
    }
}
```

Introduce Photo Validation Process

```
fun uploadImageView(app: DonationApp, imageView: ImageView) {
    val uid = app.auth.currentUser!!.uid
    val imageRef = app.storage.child("photos").child("${uid}.jpg")
    val uploadTask = imageRef.putBytes(convertImageToBytes(imageView))

    uploadTask.addOnFailureListener { object : OnFailureListener {
        override fun onFailure(error: Exception) {
            Log.v("Donation", "uploadTask.exception" + error)
        }
    }
    }.addOnSuccessListener {
        uploadTask.continueWithTask { task ->
            imageRef.downloadUrl
        }.addOnCompleteListener { task ->
            if (task.isSuccessful) {
                app.userImage = task.result!!.toString().toUri()
                updateAllDonations(app)
                writeImageRef(app,app.userImage.toString())
                Picasso.get().load(app.userImage)
                    .resize(180, 180)
                    .transform(CropCircleTransformation())
                    .into(imageView)
            }
        }
    }
}
```

Introduce Photo Validation Process



```
fun updateAllDonations(app: DonationApp) {
    val userId = app.auth.currentUser!!.uid
    val userEmail = app.auth.currentUser!!.email
    val donationRef = app.database.ref.child("donations")
                                     .orderByChild("email")
    val userdonationRef = app.database.ref.child("user-donations")
                                     .child(userId).orderByChild("uid")

    donationRef.equalTo(userEmail).addListenerForSingleValueEvent(
        object : ValueEventListener {
        override fun onCancelled(error: DatabaseError) {}
        override fun onDataChange(snapshot: DataSnapshot) {
            snapshot.children.forEach {
                it.ref.child("profilepic")
                    .setValue(app.userImage.toString())
            }
        }
    })
    userdonationRef.addListenerForSingleValueEvent(
        object : ValueEventListener {
        override fun onCancelled(error: DatabaseError) {}
        override fun onDataChange(snapshot: DataSnapshot) {
            snapshot.children.forEach {
                it.ref.child("profilepic")
                    .setValue(app.userImage.toString())
            }
        }
    })
    writeImageRef(app, app.userImage.toString())
}
```

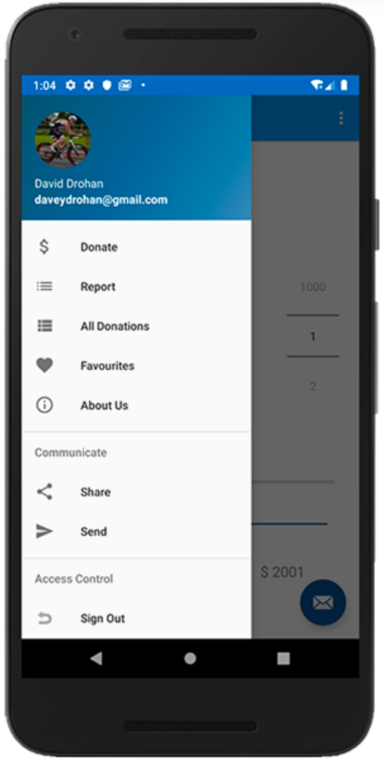
Introduce Photo Validation Process

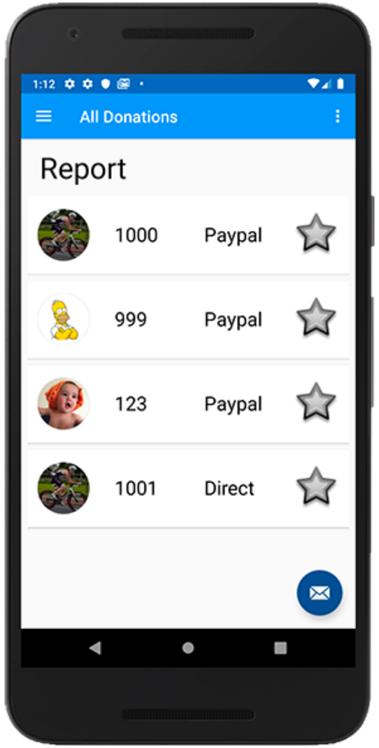
```kotlin
fun writeImageRef(app: DonationApp, imageRef: String) {
    val userId = app.auth.currentUser!!.uid
    val values = UserPhotoModel(userId,imageRef).toMap()
    val childUpdates = HashMap<String, Any>()

    childUpdates["/user-photos/$userId"] = values
    app.database.updateChildren(childUpdates)
}
```

Respond to ImageView Click in Nav Drawer

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    when (requestCode) {
        1 -> {
            if (data != null) {
                writeImageRef(app,readImageUri(resultCode, data).toString())
                Picasso.get().load(readImageUri(resultCode, data).toString())
                    .resize(180, 180)
                    .transform(CropCircleTransformation())
                    .into(navView.getHeaderView(0).imageView, object : Callback {
                        override fun onSuccess() {
                            // Drawable is ready
                            uploadImageView(app,navView.getHeaderView(0).imageView)
                        }
                        override fun onError(e: Exception) {}
                    })
            }
        }
    }
}
```

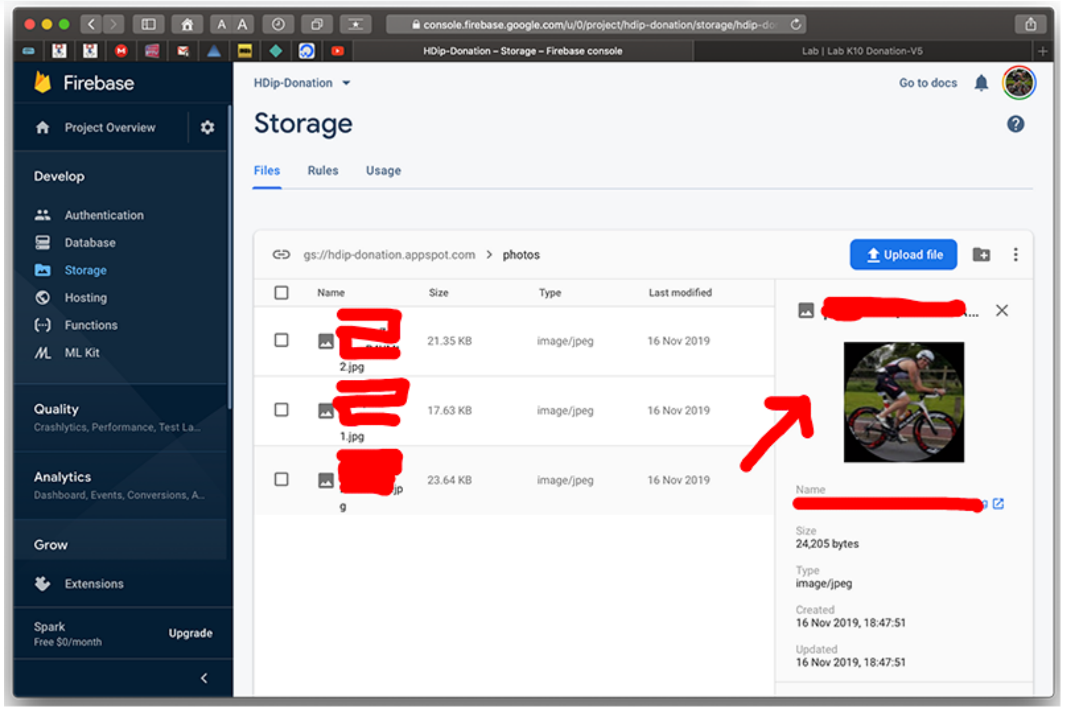
Cloud Storage + Mobile App





Report

1000	Paypal	☆
999	Paypal	☆
123	Paypal	☆
1001	Direct	☆



console.firebase.google.com/u/0/project/hdip-donation/storage/hdip-do
HDip-Donation – Storage – Firebase console
Lab | Lab K10 Donation-V5
Firebase
Project Overview
Develop
Authentication
Database
Storage
Hosting
Functions
ML Kit
Quality
Crashlytics, Performance, Test La...
Analytics
Dashboard, Events, Conversions, A...
Grow
Extensions
Spark
Free $0/month
Upgrade
HDip-Donation
Go to docs
Files
Rules
Usage
Upload file

Storage

gs://hdip-donation.appspot.com > photos

Name	Size	Type	Last modified
[redacted] 2.jpg	21.35 KB	image/jpeg	16 Nov 2019
[redacted] 1.jpg	17.63 KB	image/jpeg	16 Nov 2019
[redacted]	23.64 KB	image/jpeg	16 Nov 2019

[redacted]

Name
[redacted]

Size
24,205 bytes

Type
image/jpeg

Created
16 Nov 2019, 18:47:51

Updated
16 Nov 2019, 18:47:51

References

https://console.firebase.google.com/
https://firebase.google.com/
https://github.com/firebase/quickstart-android

Thanks.