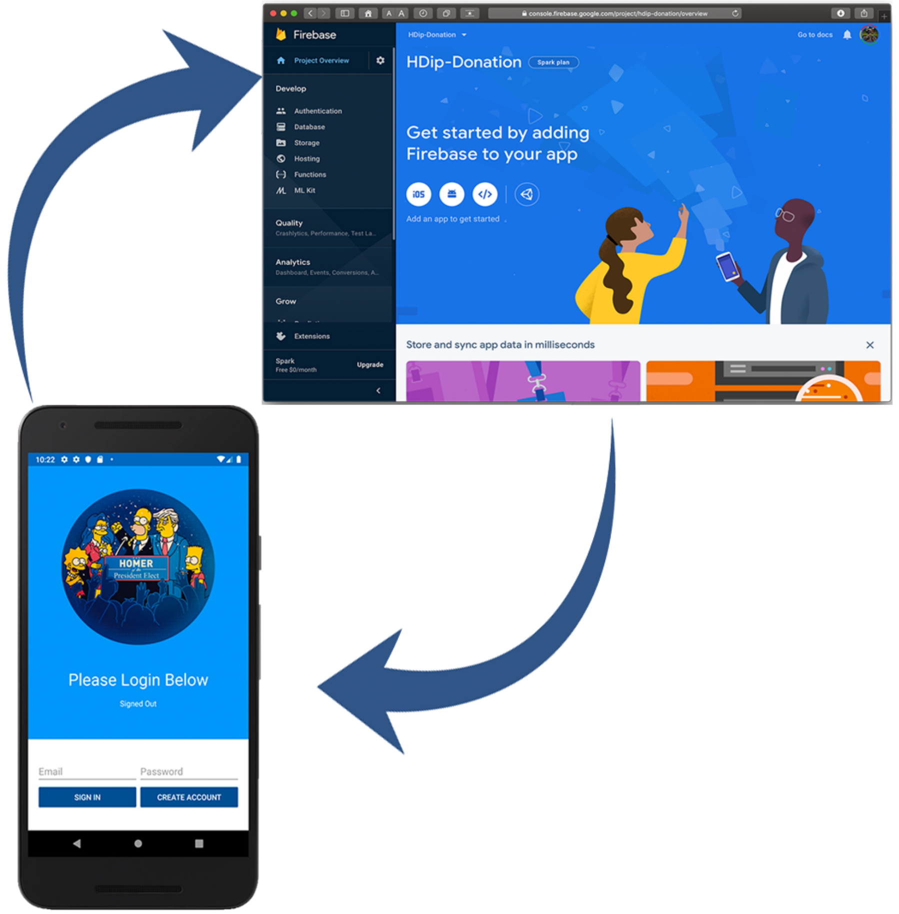
Mobile Application Development

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-V4 Walk through

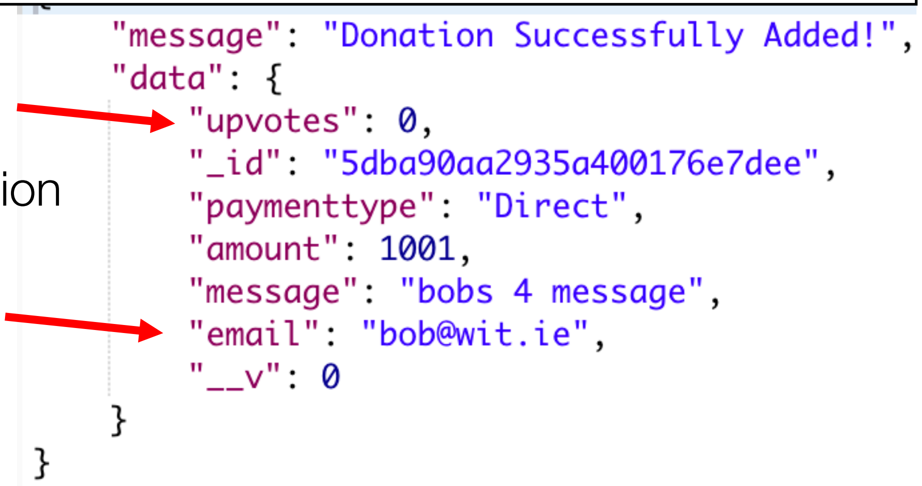


Aside – Multi User Donation Service

- ❑ http://donationweb-hdip-mu-server.herokuapp.com api endpoints

```
{ method: 'GET', path: ‘/donations/', config: Donations.getAll },
{ method: 'GET', path: ‘/donations/:email', config: Donations.findAll },
{ method: 'GET', path: '/donations/:email/{id}', config: Donations.findOne },
{ method: 'POST', path: '/donations/:email', config: Donations.addDonation },
{ method: 'PUT', path: '/donations/:email/{id}', config: Donations.editDonation },
{ method: 'DELETE', path: /donations/:email/{id}', config: Donations.deleteDonation }
```

- ❑ Use **DonationService** for
  - Adding / Updating / Deleting a Donation
  - Listing All Donations
  - Finding a single Donation
- ❑ FOR A **SINGLE USER**



```
    "message": "Donation Successfully Added!",
    "data": {
        "upvotes": 0,
        "_id": "5dba90aa2935a400176e7dee",
        "paymenttype": "Direct",
        "amount": 1001,
        "message": "bobs 4 message",
        "email": "bob@wit.ie",
        "__v": 0
    }
}
```

Steps to integrate Firebase Auth into your App

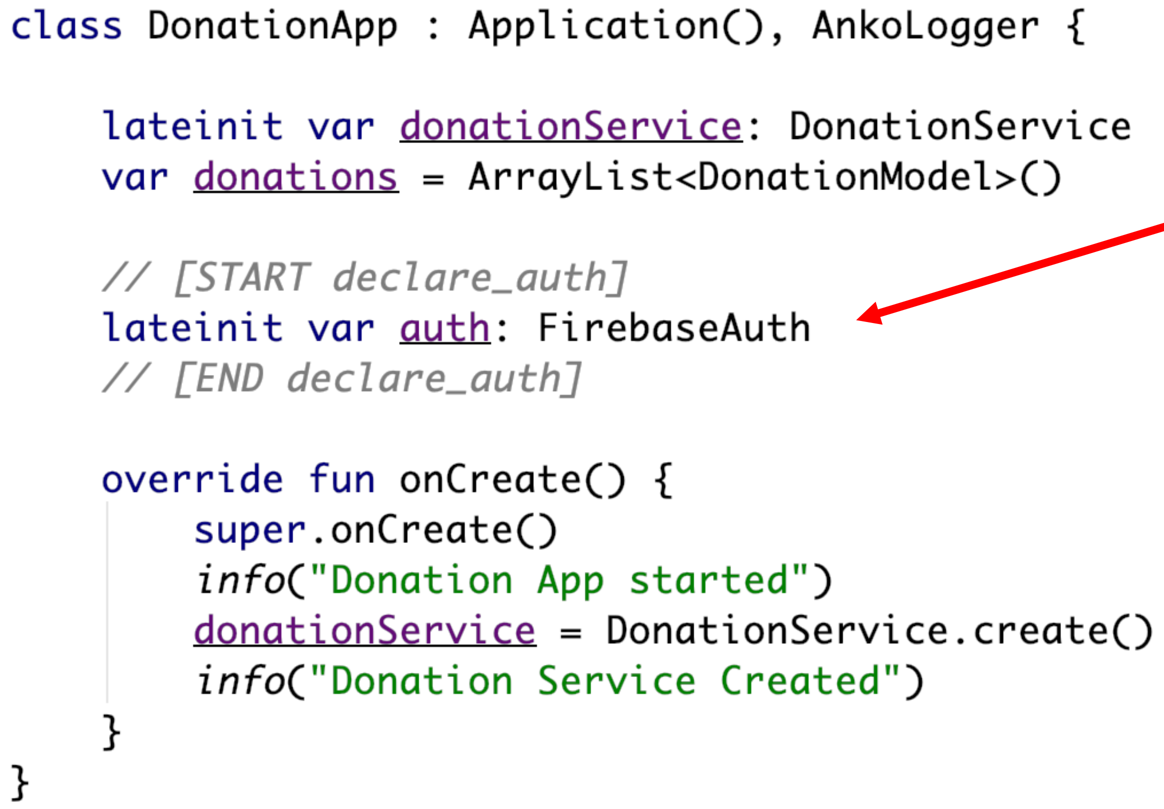
1. Create your Firebase Project
2. Setup your Sign-In Method
3. Setup Firebase in your App
4. Introduce Authentication Flow – Create Account & Sign In / Sign Out
5. Utilize your Authenticated User in your App

Steps to integrate Firebase Auth into your App

1. Create your Firebase Project
2. Setup your Sign-In Method
3. Setup Firebase in your App
4. Introduce Authentication Flow – Create Account & Sign In / Sign Out
5. Utilize your Authenticated User in your App

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

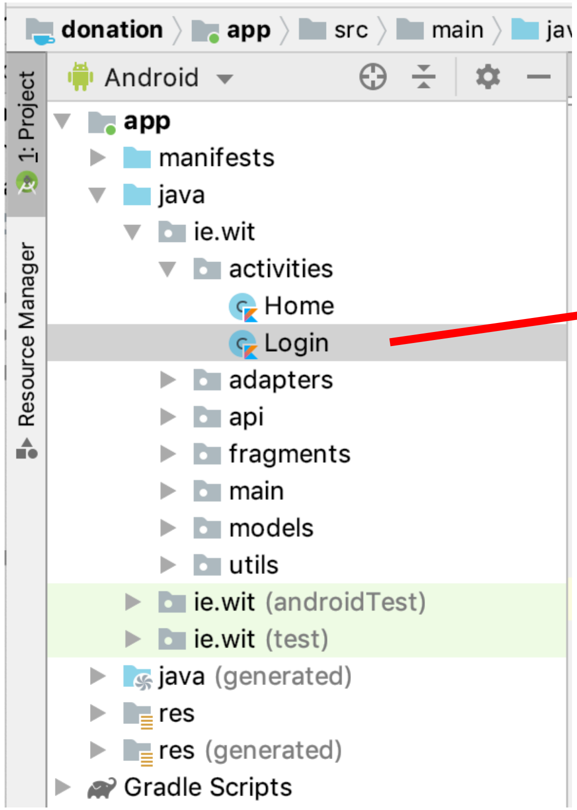
Introduce Authentication Flow – Create Account



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

    lateinit var donationService: DonationService
    var donations = ArrayList<DonationModel>()

    // [START declare_auth]
    lateinit var auth: FirebaseAuth
    // [END declare_auth]

    override fun onCreate() {
        super.onCreate()
        info("Donation App started")
        donationService = DonationService.create()
        info("Donation Service Created")
    }
}
```

Our **FirebaseAuth** instance

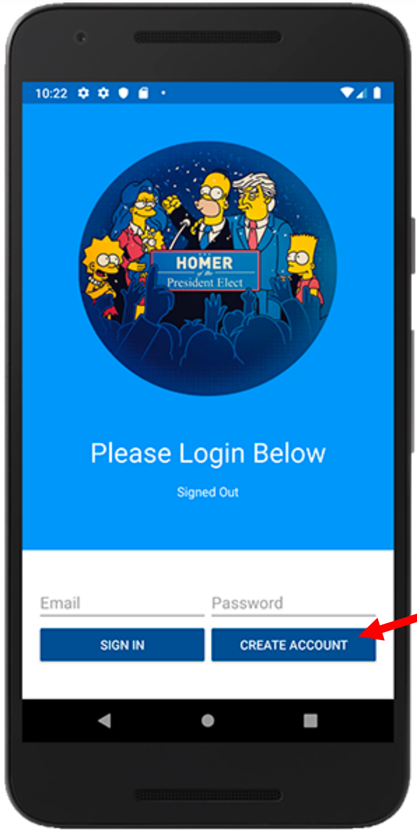
Introduce Authentication Flow – Create Account



donation 〉 app 〉 src 〉 main 〉 jav

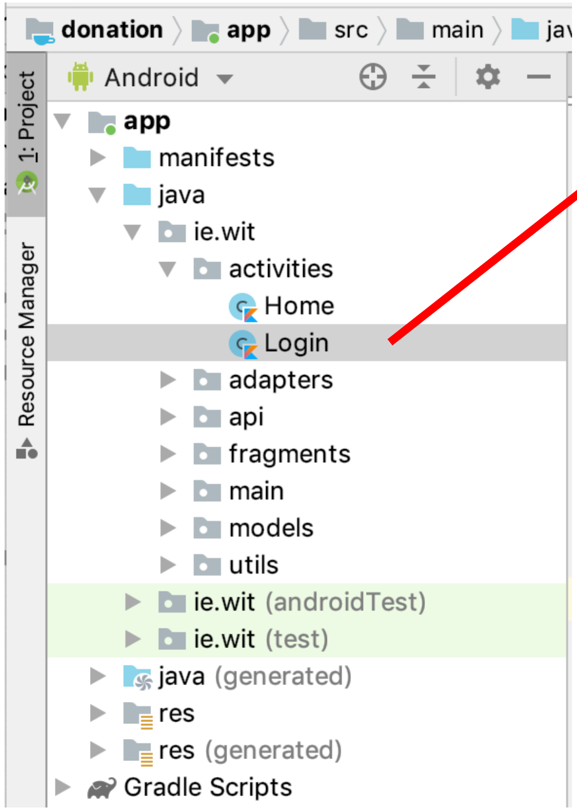
Android

- app
  - manifests
  - java
    - ie.wit
      - activities
        - Home
        - Login
      - adapters
      - api
      - fragments
      - main
      - models
      - utils
    - ie.wit (androidTest)
    - ie.wit (test)
  - java (generated)
  - res
  - res (generated)
- Gradle Scripts



Our **Login** Activity

Introduce Authentication Flow – Create Account



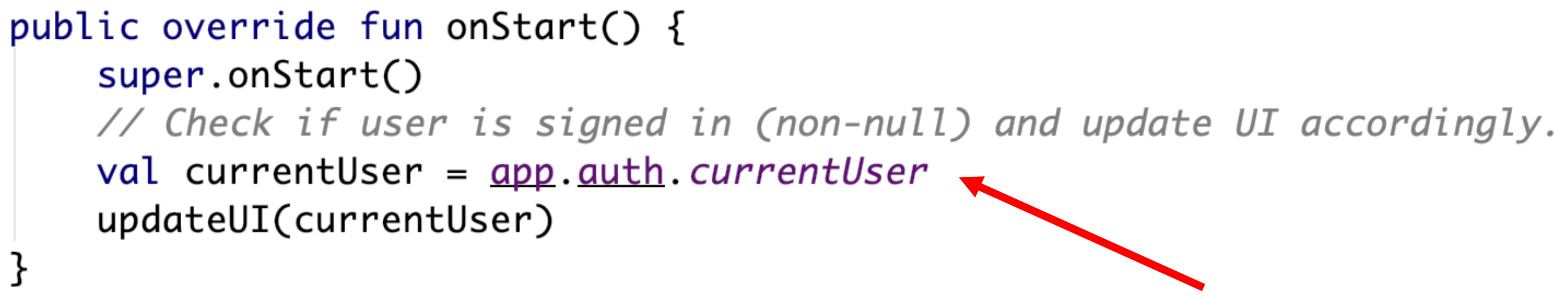
```
class Login : AppCompatActivity(), View.OnClickListener {

    lateinit var app: DonationApp
    lateinit var loader : AlertDialog

    public override fun onCreate(savedInstanceState: Bundle?) {...}

    public override fun onStart() {...}

    private fun createAccount(email: String, password: String) {...}

    private fun signIn(email: String, password: String) {...}

    private fun signOut() {...}

    private fun sendEmailVerification() {...}

    private fun validateForm(): Boolean {...}

    private fun updateUI(user: FirebaseUser?) {...}

    override fun onClick(v: View) {...}

    companion object {...}
}
```

Our **Login** Activity

Introduce Authentication Flow – Create Account

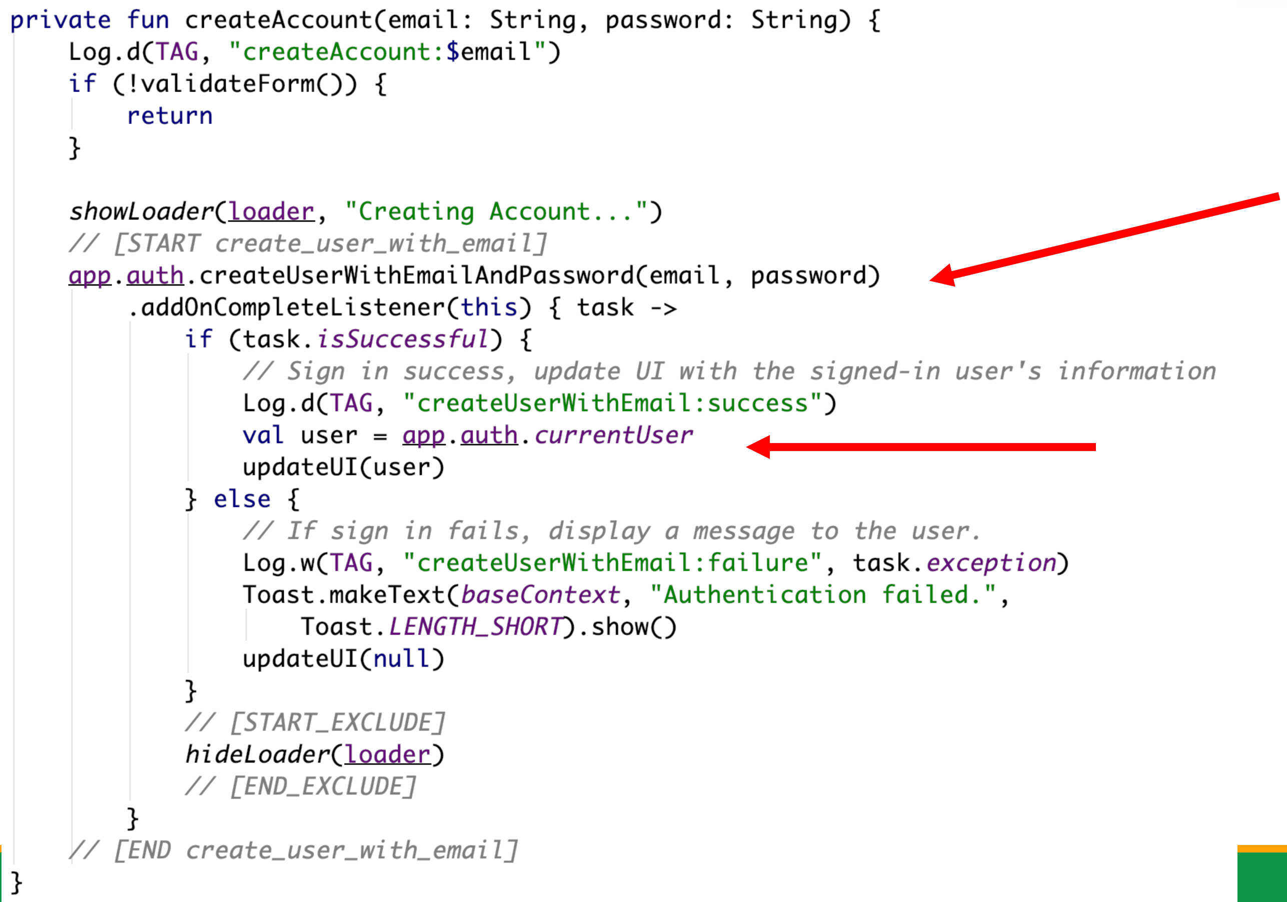
```kotlin
public override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.login)
    app = application as DonationApp
    // Buttons
    emailSignInButton.setOnClickListener(this)
    emailCreateAccountButton.setOnClickListener(this)
    signOutButton.setOnClickListener(this)
    verifyEmailButton.setOnClickListener(this)

    // [START initialize_auth]
    // Initialize Firebase Auth
    app.auth = FirebaseAuth.getInstance()
    // [END initialize_auth]

    loader = createLoader(this)
}
```

Introduce Authentication Flow – Create Account



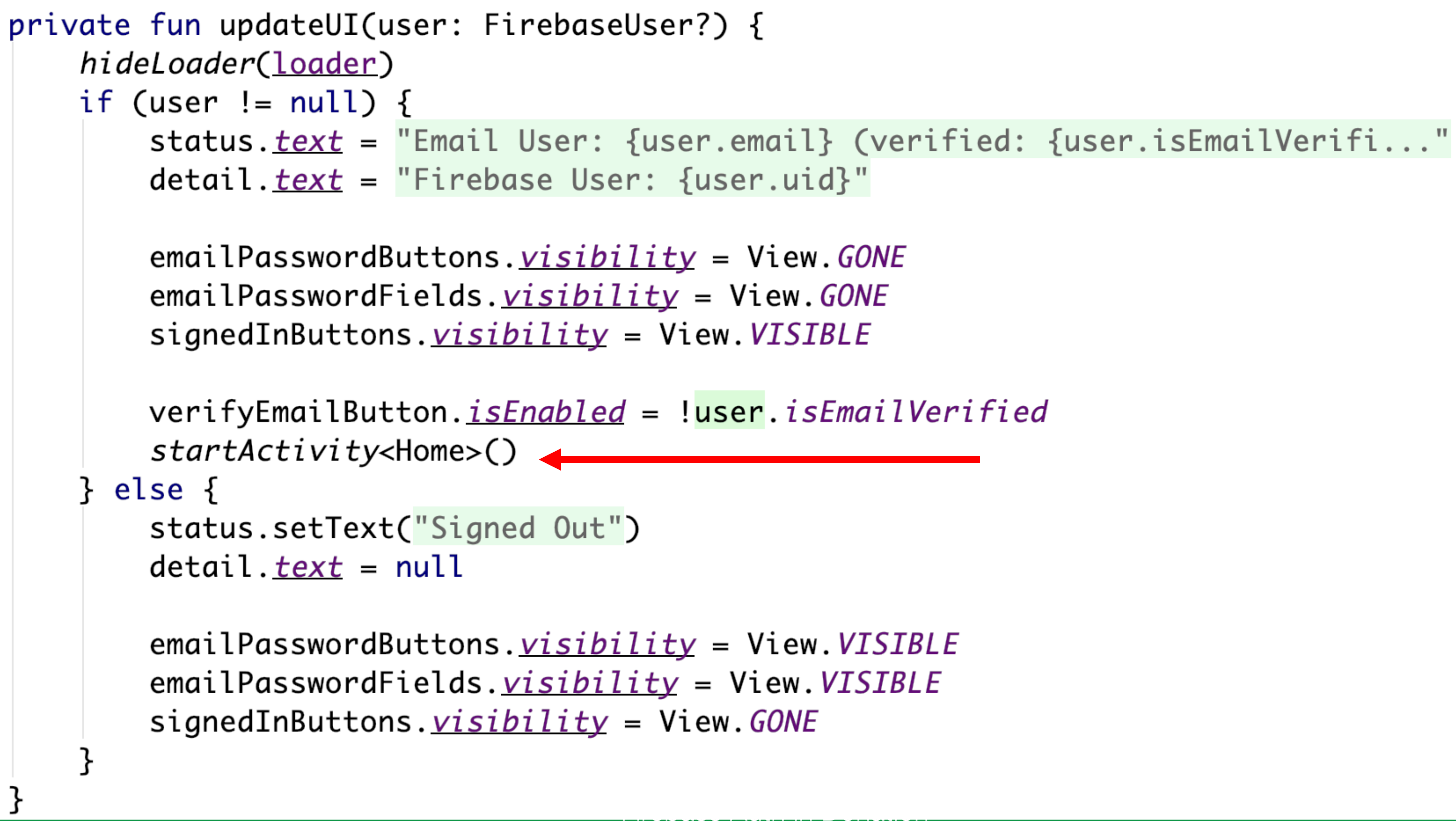
```
public override fun onStart() {
    super.onStart()
    // Check if user is signed in (non-null) and update UI accordingly.
    val currentUser = app.auth.currentUser
    updateUI(currentUser)
}
```

Introduce Authentication Flow – Create Account



```kotlin
private fun createAccount(email: String, password: String) {
    Log.d(TAG, "createAccount:$email")
    if (!validateForm()) {
        return
    }

    showLoader(loader, "Creating Account...")
    // [START create_user_with_email]
    app.auth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                // Sign in success, update UI with the signed-in user's information
                Log.d(TAG, "createUserWithEmail:success")
                val user = app.auth.currentUser
                updateUI(user)
            } else {
                // If sign in fails, display a message to the user.
                Log.w(TAG, "createUserWithEmail:failure", task.exception)
                Toast.makeText(baseContext, "Authentication failed.",
                    Toast.LENGTH_SHORT).show()
                updateUI(null)
            }
            // [START_EXCLUDE]
            hideLoader(loader)
            // [END_EXCLUDE]
        }
    // [END create_user_with_email]
}
```

Introduce Authentication Flow – Create Account



```kotlin
private fun updateUI(user: FirebaseUser?) {
    hideLoader(loader)
    if (user != null) {
        status.text = "Email User: {user.email} (verified: {user.isEmailVerifi..."
        detail.text = "Firebase User: {user.uid}"

        emailPasswordButtons.visibility = View.GONE
        emailPasswordFields.visibility = View.GONE
        signedInButtons.visibility = View.VISIBLE

        verifyEmailButton.isEnabled = !user.isEmailVerified
        startActivity<Home>()
    } else {
        status.setText("Signed Out")
        detail.text = null

        emailPasswordButtons.visibility = View.VISIBLE
        emailPasswordFields.visibility = View.VISIBLE
        signedInButtons.visibility = View.GONE
    }
}
```

Introduce Authentication Flow – Sign In User

```
private fun signIn(email: String, password: String) {
    Log.d(TAG, "signIn:$email")
    if (!validateForm()) {
        return
    }
    showLoader(loader, "Logging In...")
    // [START sign_in_with_email]
    app.auth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                // Sign in success, update UI with the signed-in user's information
                Log.d(TAG, "signInWithEmail:success")
                val user = app.auth.currentUser
                updateUI(user)
            } else {
                // If sign in fails, display a message to the user.
                Log.w(TAG, "signInWithEmail:failure", task.exception)
                Toast.makeText(baseContext, "Authentication failed.",
                    Toast.LENGTH_SHORT).show()
                updateUI(null)
            }
            // [START_EXCLUDE]
            if (!task.isSuccessful) {
                status.setText("Authentication failed")
            }
            hideLoader(loader)
            // [END_EXCLUDE]
        }
    // [END sign_in_with_email]
}
```

Utilize your Authenticated User in your App

- ❑ Setting the email in 'Home'

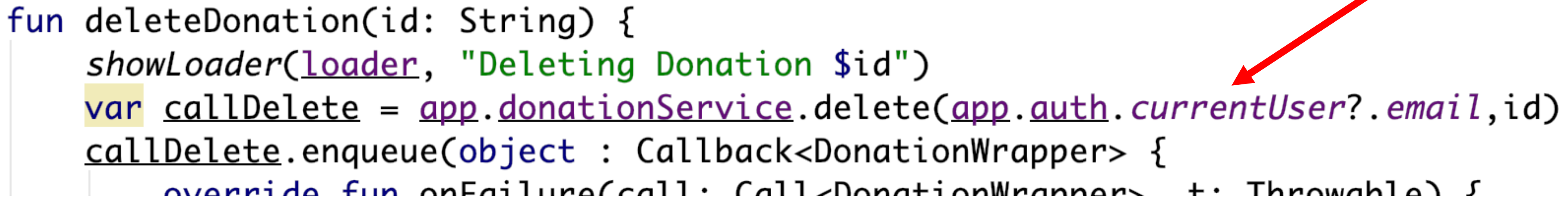
```
navView.getHeaderView(0).nav_header_email.text = app.auth.currentUser?.email
```

- ❑ Getting All Donations in 'Donate'

```
fun getAllDonations() {
    showLoader(loader, "Downloading Donations List")
    var callGetAll = app.donationService.findall(app.auth.currentUser?.email)
    callGetAll.enqueue(this)
}
```

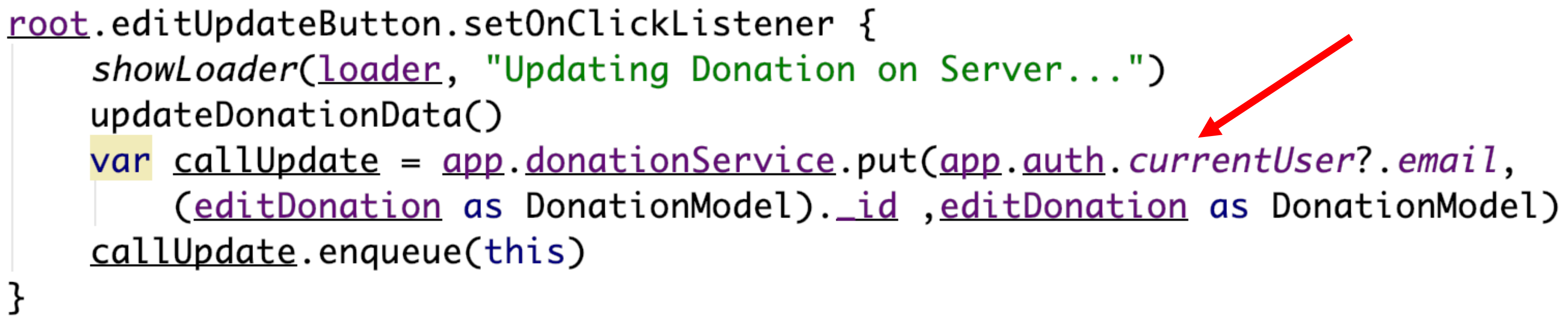
Utilize your Authenticated User in your App

- Deleting a Donation



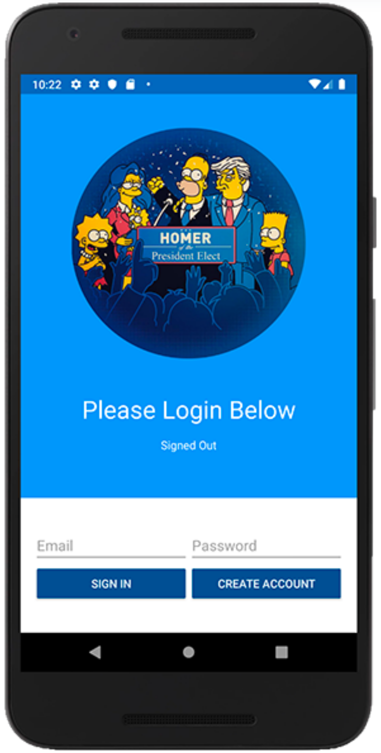
```
fun deleteDonation(id: String) {
    showLoader(loader, "Deleting Donation $id")
    var callDelete = app.donationService.delete(app.auth.currentUser?.email,id)
    callDelete.enqueue(object : Callback<DonationWrapper> {
        override fun onFailure(call: Call<DonationWrapper>, t: Throwable) {
```

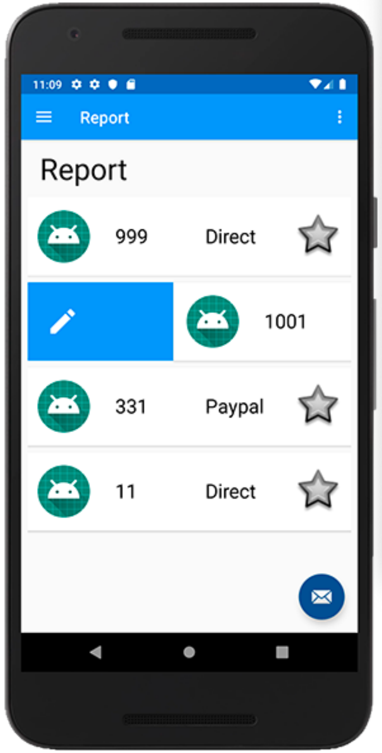
- Updating a Donation

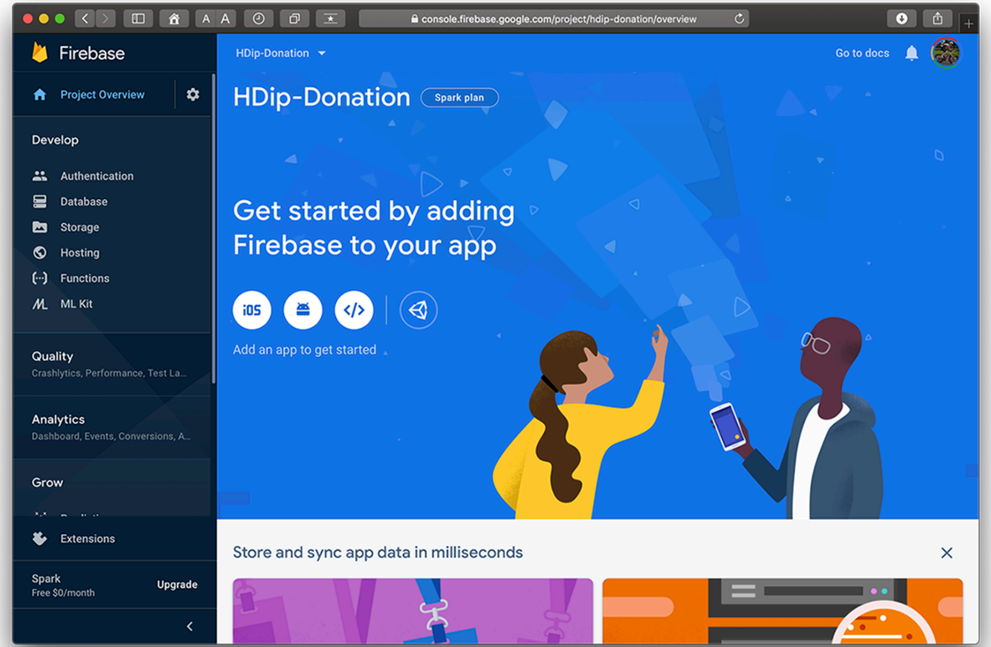


```
root.editUpdateButton.setOnClickListener {
    showLoader(loader, "Updating Donation on Server...")
    updateDonationData()
    var callUpdate = app.donationService.put(app.auth.currentUser?.email,
        (editDonation as DonationModel)._id ,editDonation as DonationModel)
    callUpdate.enqueue(this)
}
```

Donation Service + Mobile App






console.firebase.google.com/project/hdip-donation/overview
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
HDip-Donation
Spark plan
Get started by adding Firebase to your app
iOS
Add an app to get started
Store and sync app data in milliseconds

References

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

Thanks.