Code Review

Evidence of Using Secure Protocols for Front-End and Back-End Interaction

Firebase SDKs Handle SSL/TLS Automatically

- Firebase automatically secures communication using HTTPS.
- **Firebase Authentication**, **Firestore**, **Realtime Database**, **Cloud Storage**, and **Cloud Messaging** all use secure protocols by default.
- Firebase SDKs for Android ensure secure communication automatically, requiring no manual setup for SSL/TLS.

Example of secure communication:

`com.google.firebase:firebase-auth` or `com.google.firebase:firebase-firestore` libraries default to HTTPS.

```
"project_info": {
  "project_number": "35123530913",
  "firebase_url": "https://reflectionsoffaith-13291-default-rtdb.firebaseio.com",
  "project_id": "reflectionsoffaith-13291",
  "storage_bucket": "reflectionsoffaith-13291.appspot.com"
```

Basic Program Skills

Basic Concepts

- **Classes**, **Methods**, **Arrays**, and **Loops**: Each page has at least one class containing multiple methods with arrays, loops, and variables.

```
// Loop through children appending to the array until we hit depth

var i = 0;
while (options.depth > 0 && (segment = child.exec(key)) !== null && i < options.depth) {
    i += 1;
    if (!options.plainObjects && has.call(Object.prototype, segment[1].slice(1, -1))) {
        if (!options.allowPrototypes) {
            return;
        }
    }
    keys.push(segment[1]);
}
```

```
class Log_In_Screen : AppCompatActivity() {

    private lateinit var biometricPrompt: BiometricPrompt
    private lateinit var promptInfo: BiometricPrompt.PromptInfo
```

Exception Handling

- Exception handling through `.onError` handlers and `try-catch` blocks.

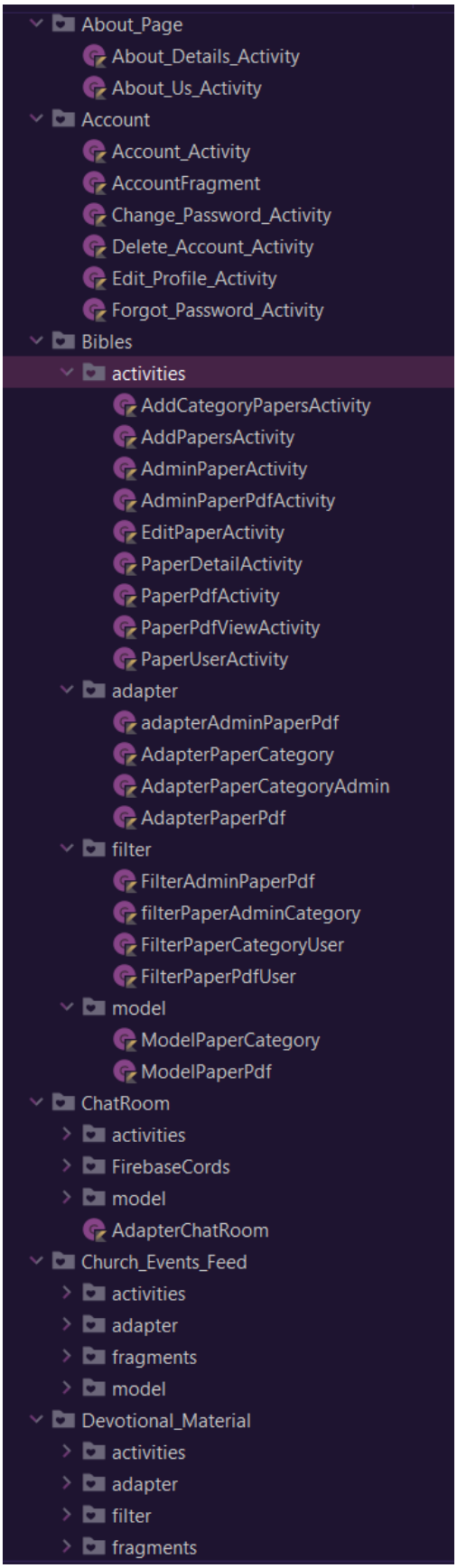
```
try {
    val account = task.getResult(ApiException::class.java)
    Log.d(TAG,  msg: "googleSignInARL: Account ID: ${account.id}")
    firebaseAuthWithGoogleAccount(account.idToken)
} catch (e: Exception) {
    Log.d(TAG,  msg: "googleSignInARL", e)
    Utils.toast( context: this@Log_In_Screen,  message: "${e.message}")
}
```

```
.onError { t →
    Log.e(TAG,  msg: "loadBookFromUrl: ${t.message}")
}
.onPageError { page, t →
    Utils.toast( context: this,  message: "Error at page: $page")
    Log.d(TAG,  msg: "loadBookFromUrl: ${t.message}")
}

.onError { error →
    error.message?.let {
        Utils.toast( context: this, it)
    }
```

Application Structure

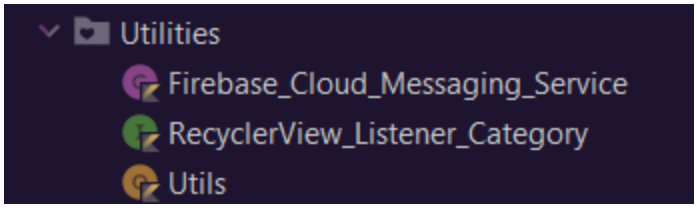
Logical Separation of Code

- Code is organized into separate classes based on functionality. This promotes modularity and maintainability.


About_Page
About_Details_Activity
About_Us_Activity
Account
Account_Activity
AccountFragment
Change_Password_Activity
Delete_Account_Activity
Edit_Profile_Activity
Forgot_Password_Activity
Bibles
activities
AddCategoryPapersActivity
AddPapersActivity
AdminPaperActivity
AdminPaperPdfActivity
EditPaperActivity
PaperDetailActivity
PaperPdfActivity
PaperPdfViewActivity
PaperUserActivity
adapter
adapterAdminPaperPdf
AdapterPaperCategory
AdapterPaperCategoryAdmin
AdapterPaperPdf
filter
FilterAdminPaperPdf
filterPaperAdminCategory
FilterPaperCategoryUser
FilterPaperPdfUser
model
ModelPaperCategory
ModelPaperPdf
ChatRoom
activities
FirebaseCords
model
AdapterChatRoom
Church_Events_Feed
activities
adapter
fragments
model
Devotional_Material
activities
adapter
filter
fragments

Utility Class

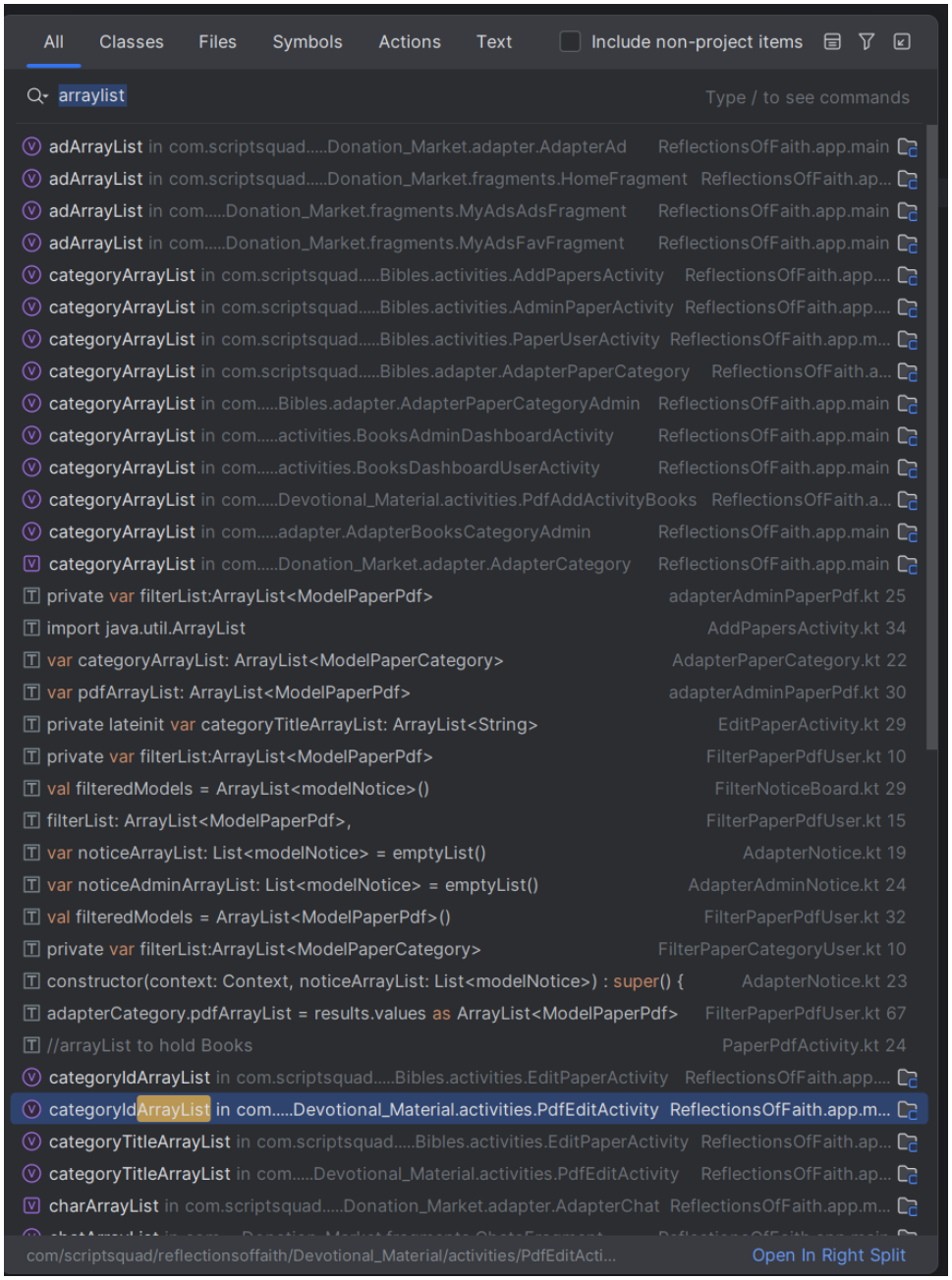
- Utility functions are kept in a dedicated class to maintain code organization.



Data Structures

- **ArrayList**
  - Demonstrates dynamic array usage.

```
constructor(
    filterList: ArrayList<ModelBooksCategoryAdmin>,
    adapterCategory: AdapterBooksCategoryAdmin
) : super() {
    this.filterList = filterList
    this.adapterCategory = adapterCategory
}
```



- **HashMap**
  - Efficient key-value pair storage for quick lookups.

```
HashMap                                                    Results might be incomplete until all dependencies are loaded
Android Lint: Performance: HashMap can be replaced with SparseArray
val hashMap = HashMap<String, Any>()                                   AddCategoryPapersActivity.kt 95
hashMap["id"] = "$timestamp"                                           AddCategoryPapersActivity.kt 96
hashMap["category"] = category                                         AddCategoryPapersActivity.kt 97
val hashMap = HashMap<String, Any>()                                   Register_Using_Email_Activity.kt 152
val hashMap = HashMap<String,Any>()                                    EditPaperActivity.kt 210
val hashMap = HashMap<String, Any>()                                   Utils.kt 113
val hashMap = HashMap<String, Any>()                                   Image_Upload_Preview_Activity.kt 158
val hashMap = HashMap<String, Any>()                                   Edit_Profile_Activity.kt 184
val hashMap: HashMap<String, Any> = HashMap()                          Admin_Notices_Activity.kt 217
val hashMap = HashMap<String, Any>()                                   Ad_Details_Activity.kt 177
hashMap["adId"] = adId                                                 Utils.kt 114
hashMap["name"] = "$name"                                              Register_Using_Email_Activity.kt 153
hashMap ["title"] = "$title"                                           EditPaperActivity.kt 211
hashMap["categoryId"] = "$selectedCategoryId"                          EditPaperActivity.kt 212
.updateChildren(hashMap)                                               EditPaperActivity.kt 218
val hashMap = HashMap<String, Any>()                                   AddCategoryPapersActivity.kt 95
val hashMap = HashMap<String, Any>()                                   Register_Using_Email_Activity.kt 152
hashMap["timestamp"] = timestamp                                       AddCategoryPapersActivity.kt 98
hashMap["uid"] = "${firebaseAuth.uid}"                                 AddCategoryPapersActivity.kt 99
.setValue(hashMap)                                                     AddCategoryPapersActivity.kt 105
val hashMap = HashMap<String, Any?>()                                  Log_In_Screen.kt 243
val hashMap = HashMap<String, Any>()                                   Ad_Details_Activity.kt 177
val hashMap = HashMap<String, Any>()                                   Edit_Profile_Activity.kt 184
val hashMap = hashMapOf<String,Any?>()                                 Log_In_Using_Phone_Activity.kt 272
val hashMap: HashMap<String, Any> = HashMap()                          AddPapersActivity.kt 333
hashMap["phoneCode"] = ""                                              Register_Using_Email_Activity.kt 154
val hashMap = HashMap<String, Any>()                                   Chat_Activity.kt 415
val hashMap: HashMap<String, Any> = HashMap()                          Admin_Notices_Activity.kt 217
val hashMap: HashMap<String, Any> = HashMap()                          Add_Video_Activity.kt 180
val hashMap = HashMap<String, Any>().apply {                           Chat_Room_Activity.kt 131
... more
```

- **LinkedList**
  - Dynamic list structure for optimal insertion/removal operations.

```java
List<View> prevViews = new LinkedList<>();
StringBuilder errorMessage = new StringBuilder();
while (selectedViewIterator.hasNext()) {
  View selectedView = selectedViewIterator.next();
  Rect viewRect = getRect(selectedView);
  if (!viewRect.isEmpty()
      && !(selectedView instanceof TextView
          && ((TextView) selectedView).getText().length() == 0)) {
    for (View prevView : prevViews) {
      // Mutual intersection of ImageViews is acceptable in most cases.
      if (selectedView instanceof ImageView && prevView instanceof ImageView) {
        continue;
      }
      Rect prevRect = getRect(prevView);
      if (Rect.intersects(viewRect, prevRect)) {
        // Overlap detected, add to the error message
        if (errorMessage.length() > 0) {
          errorMessage.append(",\n\n");
        }
        errorMessage.append(
            String.format(
                Locale.ROOT, format: "%s overlaps\n%s", describe(selectedView), describe(prevView)));
        break;
      }
    }
    prevViews.add(selectedView);
  }
}
```

Security Features

Masking of Sensitive Data

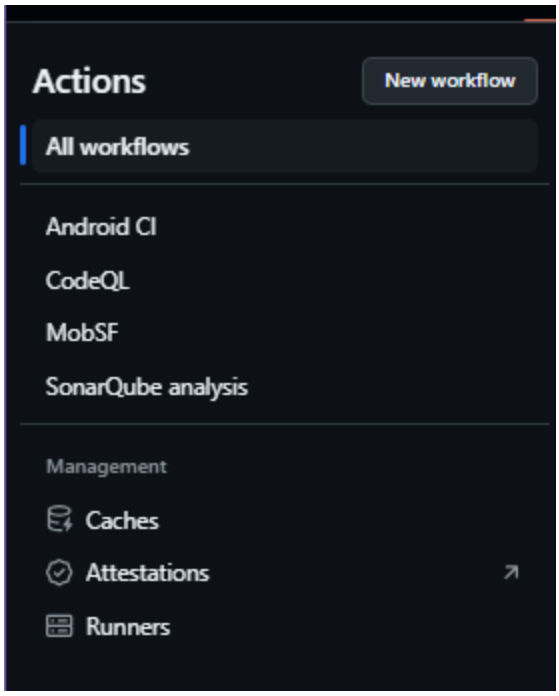
- Secure handling of sensitive user data to prevent exposure.

```kotlin
// Method to mask email (show only first and last few characters)
private fun maskEmail(email: String): String {
    val parts = email.split( ...delimiters: "@")
    if (parts.size == 2) {
        val maskedLocal = parts[0].take( n: 1) + "***" + parts[0].takeLast( n: 1)
        return "$maskedLocal@${parts[1]}"
    }
    return "Invalid Email"
}

// Method to mask passwords (replace all characters with asterisks)
private fun maskPassword(password: String): String {
    return "*".repeat(password.length)
}
```

```
private fun validData() {
    // Get user inputs and sanitize
    name = sanitizeName(binding.nameEt.text.toString().trim())
    email = sanitizeEmail(binding.emailEt.text.toString().trim())
    password = sanitizePassword(binding.passwordEt.text.toString().trim())
    cPassword = sanitizePassword(binding.ConfirmpasswordEt.text.toString().trim())

    // Mask the email and password before logging
    val maskedEmail = maskEmail(email)
    val maskedPassword = maskPassword(password)
    val maskedCPassword = maskPassword(cPassword)

    // Log the sanitized/ masked data for debugging purposes
    Log.d(TAG, msg: "validateData: email :$maskedEmail")
    Log.d(TAG, msg: "validateData: password :$maskedPassword")
    Log.d(TAG, msg: "validateData: confirmPassword :$maskedCPassword")
    // Log the user's input data for debugging purposes
```

Static Code Analysis

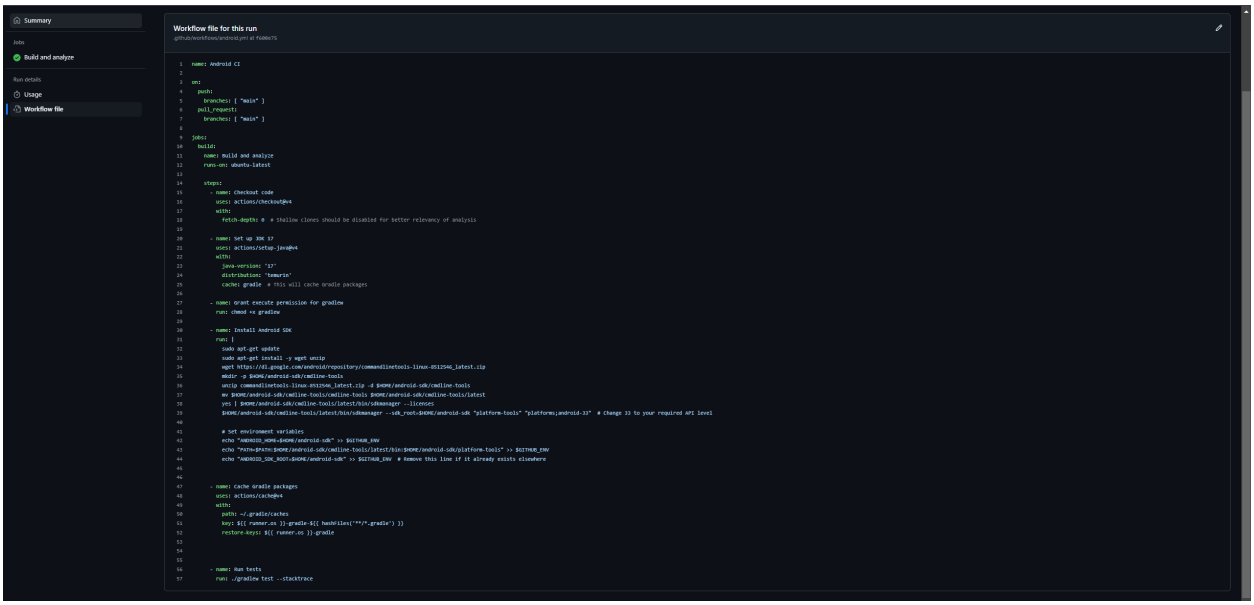
- **GitHub Actions, SonarQube, Snyk** and **MobSF** are used to ensure that the code meets security standards, with issues tracked and resolved.

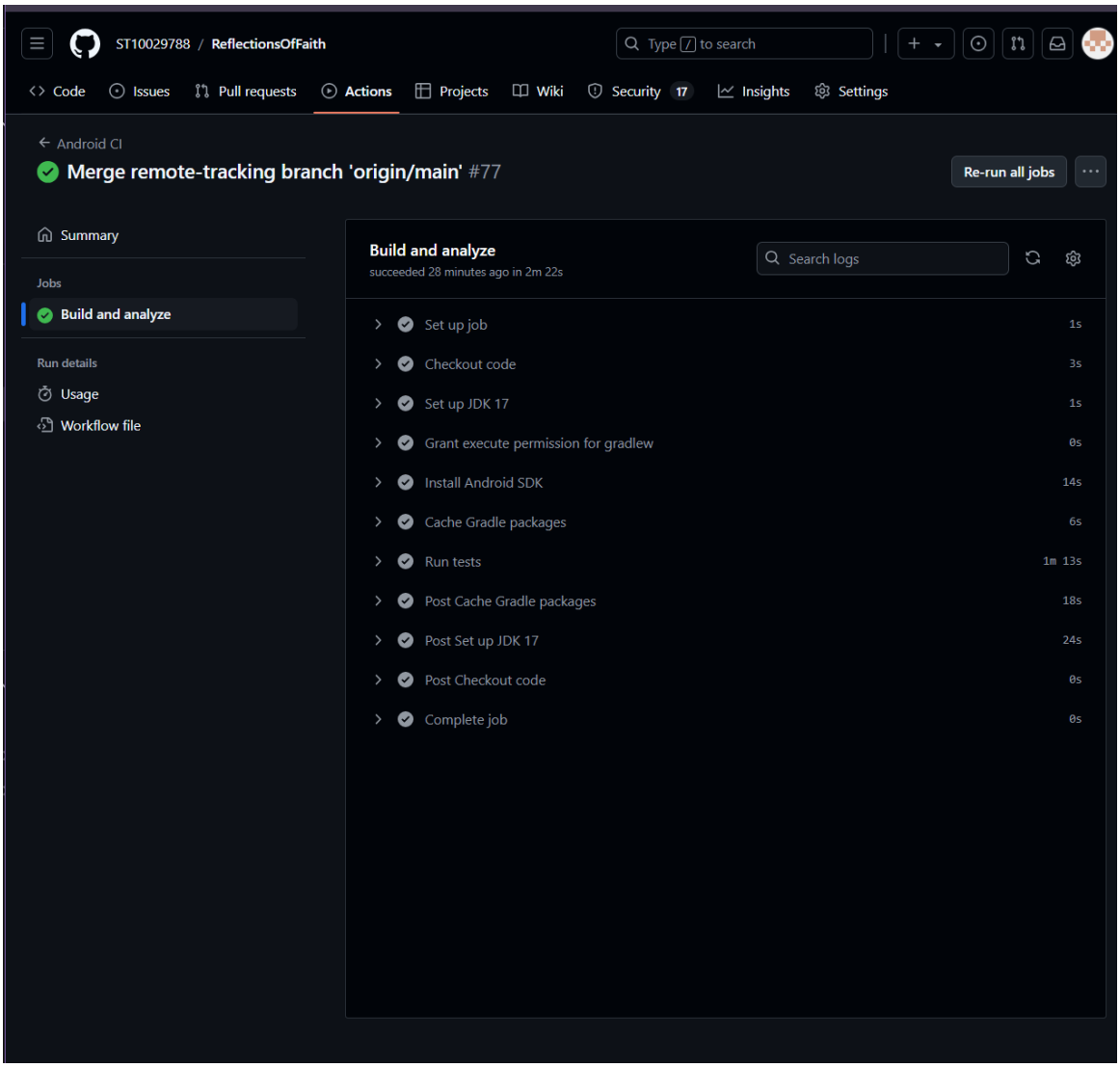


Automated Build and CI/CD

GitHub Actions Pipeline

- Automated build and deployment process implemented via a GitHub Actions pipeline.





Object-Oriented Programming

Interfaces, Encapsulation, and Polymorphism

- Interfaces, encapsulation, and polymorphism are used to achieve clean and modular code.

Interfaces Used:

```kotlin
interface RecyclerView_Listener_Category {

    fun onCategoryClick(modelCategory: ModelCategory)
}
```

Encapsulation:

```kotlin
class FilterBooksCategoryAdmin:Filter {

    private var filterList:ArrayList<ModelBooksCategoryAdmin>

    private var adapterCategory: AdapterBooksCategoryAdmin

    constructor(
        filterList: ArrayList<ModelBooksCategoryAdmin>,
        adapterCategory: AdapterBooksCategoryAdmin
    ) : super() {
        this.filterList = filterList
        this.adapterCategory = adapterCategory
    }
```

Polymorphism through Inheritance and Overriding:

```
override fun performFiltering(constraint: CharSequence?): FilterResults {                    FilterPdfUser.kt
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): HolderComments {          AdapterComment.kt
override fun onCreate(savedInstanceState: Bundle?) {                                         AddCategoryBooksActivity.kt
override fun onCreate(savedInstanceState: Bundle?) {                                         BooksDashboardUserActivity.kt
override fun onCreate(savedInstanceState: Bundle?) {                                         BookPdfAdminActivity.kt
override fun onDataChange(snapshot: DataSnapshot) {                                          BooksDashboardUserActivity.kt
override fun onCreate(savedInstanceState: Bundle?) {                                         Delete_Account_Activity.kt
override fun onCreate() {                                                                    MyApplication.kt
override fun onCreate(savedInstanceState: Bundle?) {                                         Log_In_Screen.kt
override fun onCancelled(error: DatabaseError) {                                             BooksDashboardUserActivity.kt 1
override fun performFiltering(constraint: CharSequence?): FilterResults {                    FilterPdfAdmin.kt
override fun performFiltering(constraint: CharSequence?): FilterResults {                    FilterBooksCategoryAdmin.kt
```

Chosen Programming Framework: Kotlin

- Kotlin's growing community and extensive resources were leveraged.
- Kotlin has a growing community and abundant resources, including documentation, tutorials, and open-source projects, which were used as references during devlopment.
- **Data Classes** simplify model creation and aid in the development of the **Model-View-ViewModel (MVVM)** architecture.

```
data class ModelBookPdf(
data class modelNotice(
data class modelPictures(
data class modelVideo(
```

Async Processing and Threading

- Asynchronous processing through **Runnable threads** ensures efficient background tasks.

```
mapFragment.getMapAsync( callback: this)
```

Threading

:

```
val runnable = Runnable {
    if (firebaseAuth.currentUser ≠ null
```

Testing and Localization

Testing

- **UI testing** and **Unit Testing** in `androidTest` folders ensure code reliability.

Files

main

Go to file

- .github/workflows
  - android.yml
  - mobsf.yml
  - sonarqube.yml
- .gradle
- .idea
- .kotlin
- app
  - src
    - androidTest/java/com/scripts...
      - ActivityUITest.kt
      - EditProfileActivityUITest.kt
      - ExampleInstrumentedTest.kt
      - PaperPdfViewActivityUITest.kt
    - main
    - test/java/com/scriptsquad/re...
      - UnitTest.kt
  - .gitignore
  - build.gradle
  - google-services.json
  - jitpack.properties
  - lint-baseline.xml
  - proguard-rules.pro
- build
- gradle
- README.md
- build.gradle
- gradle.properties
- gradlew
- gradlew.bat
- local.properties
- settings.gradle

ST10030992 Change Password Activity UI Test

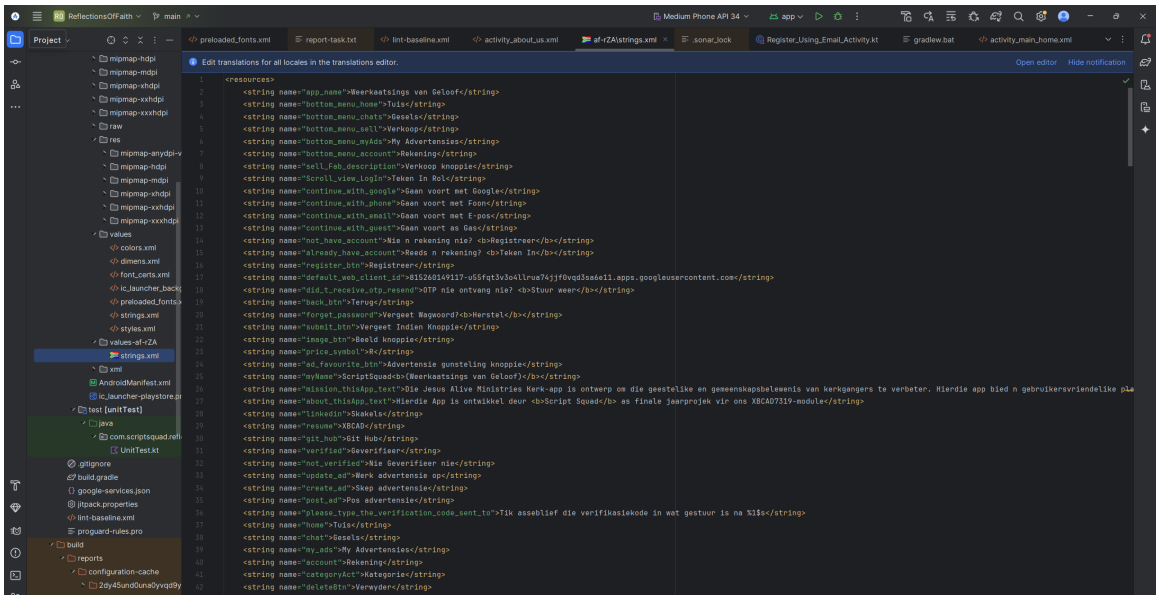
598ebea · 2 days ago History

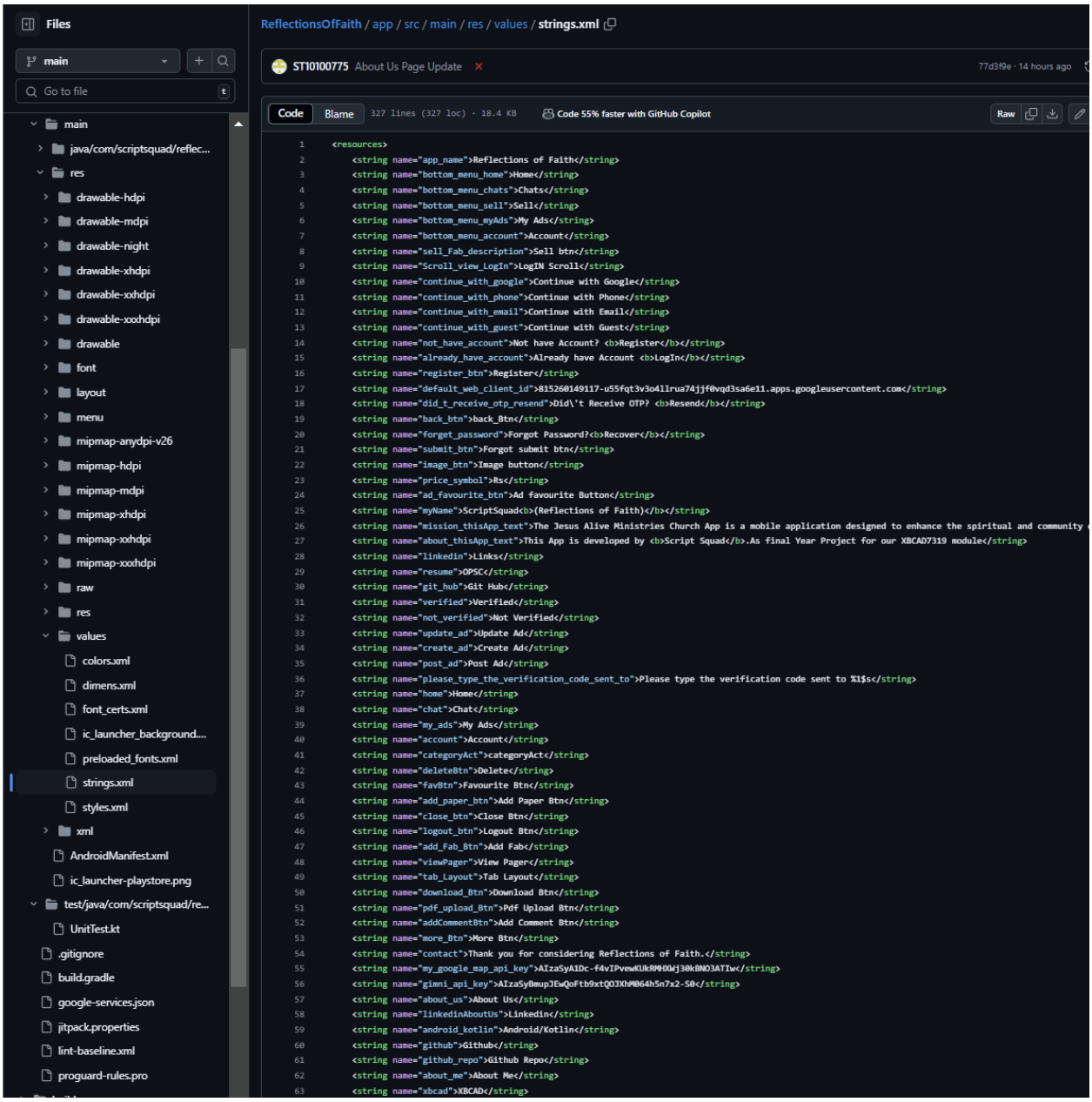
Code Blame 73 lines (72 loc) · 2.58 KB Code 55% faster with GitHub Copilot Raw

```
package com.scriptsquad.reflectionsoffaith

//
//@RunWith(MockitoJUnitRunner::class)
//class AccountActivityTest {
//
//    private lateinit var accountActivity: Account_Activity
//    private lateinit var binding: ActivityAccountBinding
//    private lateinit var firebaseAuth: FirebaseAuth
//    private lateinit var progressDialog: ProgressDialog
//
//    @Before
//    fun setUp() {
//        accountActivity = Account_Activity()
//        binding = mock(ActivityAccountBinding::class.java)
//        firebaseAuth = mock(FirebaseAuth::class.java)
//        progressDialog = mock(ProgressDialog::class.java)
//
//        // Mocking Firebase Auth instance
//        `when`(firebaseAuth.currentUser).thenReturn(mock())
//        `when`(accountActivity.firebaseAuth).thenReturn(firebaseAuth)
//
//        // Simulating Activity Creation
//        accountActivity.onCreate(mock(Bundle::class.java))
//        accountActivity.binding = binding
//    }
//
//    @Test
//    fun testDarkModeSwitchEnabled() {
//        val sharedPreferences = ApplicationProvider.getApplicationContext<Context>()
//            .getSharedPreferences("Settings", Context.MODE_PRIVATE)
//        val editor = sharedPreferences.edit()
//        editor.putBoolean("isDarkModeOn", true)
//        editor.apply()
//
//        accountActivity.onCreate(mock(Bundle::class.java))
//
//        assert(binding.darkModeSwitch.isChecked)
//        assert(AppCompatDelegate.getDefaultNightMode() == AppCompatDelegate.MODE_NIGHT_YES)
//    }
//
//    @Test
//    fun testLogout() {
//        accountActivity.logoutCv.performClick()
//
//        verify(firebaseAuth).signOut()
//        val intentCaptor = ArgumentCaptor.forClass(Intent::class.java)
//        verify(accountActivity).startActivity(intentCaptor.capture())
//        assert(intentCaptor.value.component?.className == Log_In_Screen::class.java.name)
//        verify(accountActivity).finish()
//    }
//
//    @Test
//    fun testVerifyAccount() {
//        accountActivity.verifyAccount()
//
//        verify(progressDialog).setMessage("Sending verification link to email:")
//        verify(progressDialog).show()
//
//        // Mock the behavior of sendEmailVerification and check for success
//        val user = mock(FirebaseUser::class.java)
//        `when`(firebaseAuth.currentUser).thenReturn(user)
//        `when`(user.sendEmailVerification()).thenReturn(mock(Task::class.java))
//
```

Localization

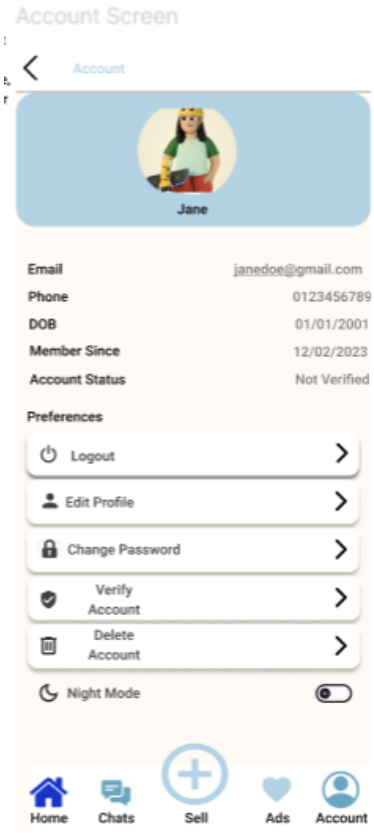
- All strings are contained in the `res` folder, with translations for **Afrikaans** stored in a separate string folder.
-

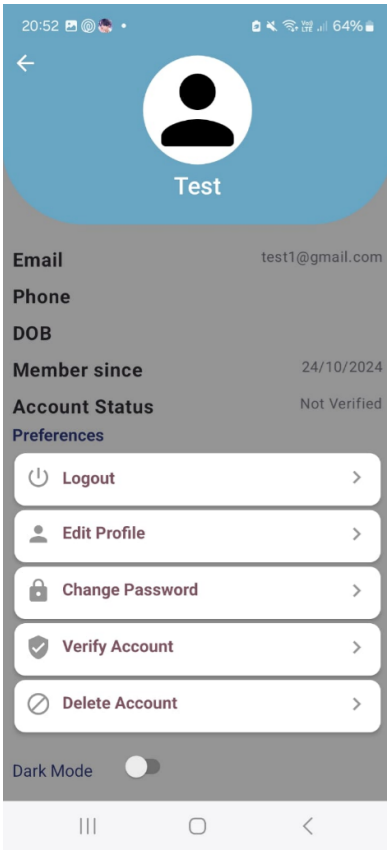

ReflectionsOfFaith
main
Medium Phone API 34
app
Project
preloaded_fonts.xml
report-task.txt
lint-baseline.xml
activity_about_us.xml
af-rZA\strings.xml
.sonar_lock
Register_Using_Email_Activity.kt
gradlew.bat
activity_main_home.xml
Edit translations for all locales in the translations editor.
Open editor
Hide notification
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
mipmap-xxxhdpi
raw
res
values
colors.xml
dimens.xml
font_certs.xml
strings.xml
styles.xml
values-af-rZA
strings.xml
xml
AndroidManifest.xml
test [unitTest]
java
UnitTest.kt
.gitignore
build.gradle
google-services.json
jitpack.properties
lint-baseline.xml
proguard-rules.pro
build
reports
configuration-cache
<resources>
<string name="app_name">Weerkaatsings van Geloof</string>
<string name="bottom_menu_home">Tuis</string>
<string name="bottom_menu_chats">Gesels</string>
<string name="bottom_menu_sell">Verkoop</string>
<string name="bottom_menu_myAds">My Advertensies</string>
<string name="bottom_menu_account">Rekening</string>
<string name="sell_Fab_description">Verkoop knoppie</string>
<string name="Scroll_view_LogIn">Teken In Rol</string>
<string name="continue_with_google">Gaan voort met Google</string>
<string name="continue_with_phone">Gaan voort met Foon</string>
<string name="continue_with_email">Gaan voort met E-pos</string>
<string name="continue_with_guest">Gaan voort as Gas</string>
<string name="not_have_account">Nie n rekening nie? Registreer</string>
<string name="already_have_account">Reeds n rekening? Teken In</string>
<string name="register_btn">Registreer</string>
<string name="default_web_client_id">815260149117-u55fqt3v3o4llrua74jjf0vqd3sa6e11.apps.googleusercontent.com</string>
<string name="did_t_receive_otp_resend">OTP nie ontvang nie? Stuur weer</string>
<string name="back_btn">Terugg</string>
<string name="forget_password">Vergeet Wagwoord?Herstal</string>
<string name="submit_btn">Vergeet Indien Knoppie</string>
<string name="image_btn">Beeld knoppie</string>
<string name="price_symbol">R</string>
<string name="ad_favourite_btn">Advertensie gunsteling knoppie</string>
<string name="myName">ScriptSquad(Weerkaatsings van Geloof)</string>
<string name="mission_thisApp_text">Die Jesus Alive Ministries Kerk-app is ontwerp om die geestelike en gemeenskapsbelewenis van kerkgangers te verbeter. Hierdie app bied n gebruikersvriendelike pla
<string name="about_thisApp_text">Hierdie App is ontwikkel deur Script Squad as finale jaarprojek vir ons XBCAD7319-module</string>
<string name="linkedin">Skakels</string>
<string name="resume">XBCAD</string>
<string name="git_hub">Git Hub</string>
<string name="verified">Geverifieer</string>
<string name="not_verified">Nie Geverifieer nie</string>
<string name="update_ad">Werk advertensie op</string>
<string name="create_ad">Skep advertensie</string>
<string name="post_ad">Pos advertensie</string>
<string name="please_type_the_verification_code_sent_to">Tik asseblief die verifikasiekode in wat gestuur is na %1$s</string>
<string name="home">Tuis</string>
<string name="chat">Gesels</string>
<string name="my_ads">My Advertensies</string>
<string name="account">Rekening</string>
<string name="categoryAct">Kategorie</string>
<string name="deleteBtn">Verwyder</string>

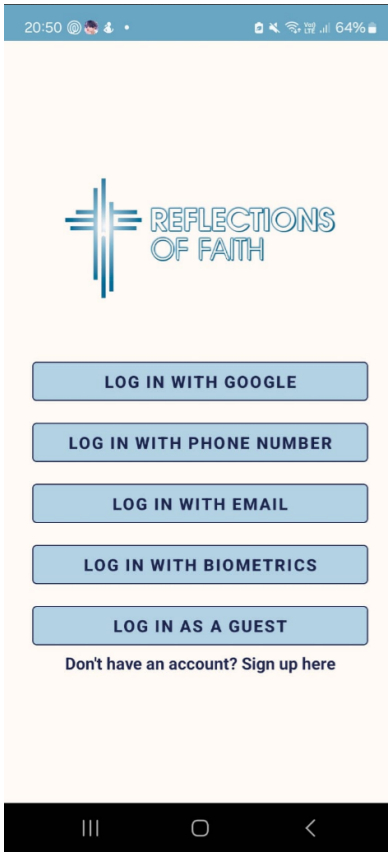


User Interface

- UI design is based on **Figma Wireframes**.


Account Screen
Account
Jane
Email
janedoe@gmail.com
Phone
0123456789
DOB
01/01/2001
Member Since
12/02/2023
Account Status
Not Verified
Preferences
Logout
Edit Profile
Change Password
Verify Account
Delete Account
Night Mode
Home
Chats
Sell
Ads
Account


20:52
64%
Test
Email
test1@gmail.com
Phone
DOB
Member since
24/10/2024
Account Status
Not Verified
Preferences
Logout
Edit Profile
Change Password
Verify Account
Delete Account
Dark Mode



Third-Party Integration

Google Maps, PDF Viewers, etc.

Firebase Services

1. **Firebase Authentication** - Manages user authentication.
2. **Firebase Realtime Database** - Provides cloud-hosted, real-time data syncing.
3. **Firebase Firestore** - A flexible, scalable NoSQL cloud database.
4. **Firebase Storage** - Manages user-generated content like photos and videos.
5. **Firebase Crashlytics** - Tracks and reports app crashes.
6. **Firebase Analytics** - Collects app usage data for insights.
7. **Firebase Cloud Messaging** - Allows push notifications.

8. **Firebase UI** (`firebase-ui-database` and `firebase-ui-firestore`) - Provides pre-built UI components for Firebase.

UI and UX Libraries

1. **Material Components** (`com.google.android.material:material`) - Implements Material Design components.
2. **Glide** - A fast image loading and caching library.
3. **MotionToast** (`com.github.Spikeysanju:MotionToast`) - Creates customizable, animated toasts.
4. **Lottie** - Displays JSON-based animations created in Adobe After Effects.
5. **Android AdvancedWebView** - Provides an extended WebView component.
6. **Country Code Picker** (`com.hbb20:ccp`) - Allows easy selection of country codes.
7. **Android PDF Viewer** (`com.github.barteksc:android-pdf-viewer`) - Displays PDFs in-app.
8. **Rounded CardView** (`com.github.captain-miao:optroundcardview`) - Provides rounded CardViews.
9. **Image Cropper** (`com.vanniktech:android-image-cropper`) - Enables image cropping in-app.

Google Services

1. **Google Play Services Authentication** - Enables Google sign-in.
2. **Google Maps & Places API** - Provides Google Maps, Places, and related utilities for location-based functionality.
3. **ExoPlayer** (`exoplayer-core` and `exoplayer-ui`) - Used for playing audio and video.

Data Processing

1. **Gson** (`com.google.code.gson`) - A library for JSON serialization and deserialization.

Biometric Authentication

1. **AndroidX Biometric** - Supports biometric authentication like fingerprint and face recognition.

Testing Libraries

1. **JUnit** - A testing framework for writing unit tests.
2. **Espresso** - A library for UI testing.
3. **Mockito** - A framework for creating mocks in unit tests.

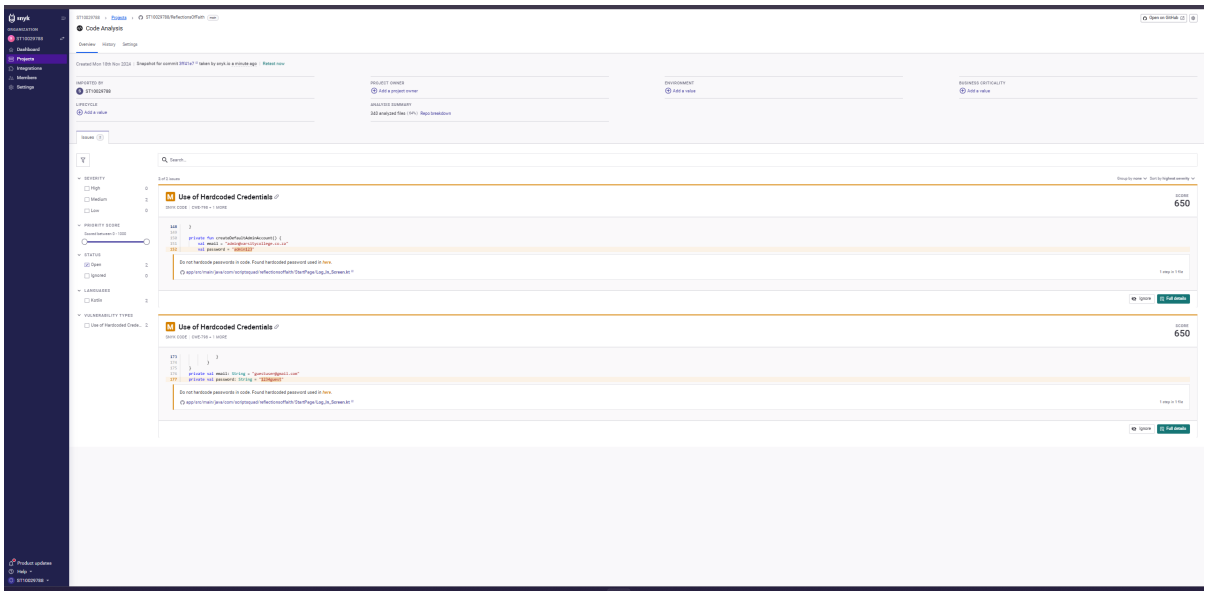
Sanitizing User Inputs

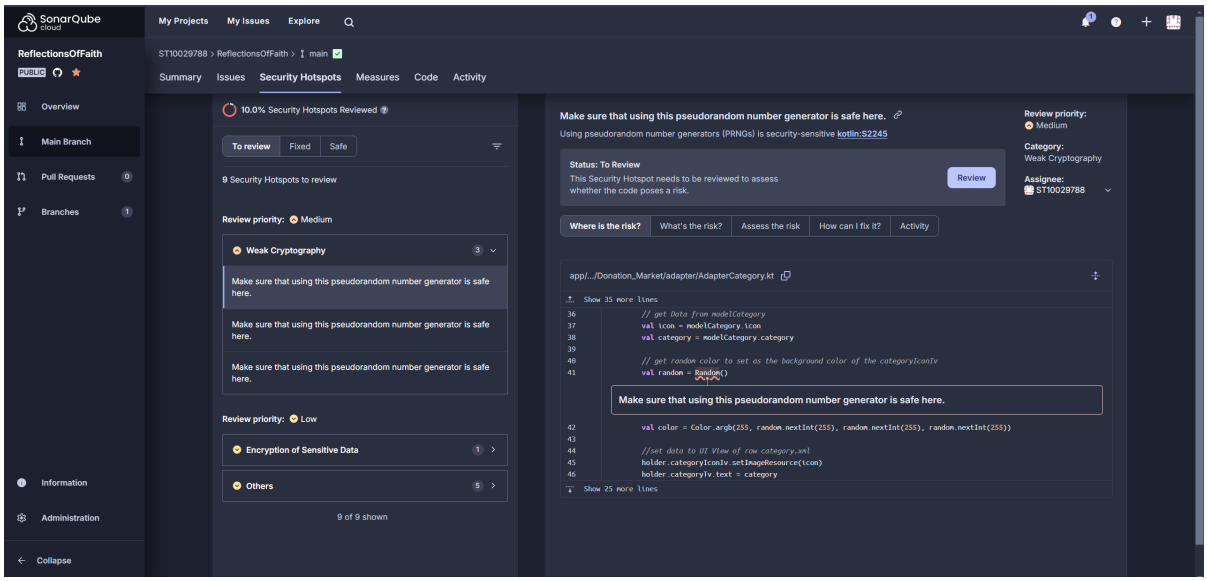
```
sanitize                                                                    Type / to see commands
// If the data is valid, sanitize inputs before login                       Log_In_Using_Email_Activity.kt 84
// Get user inputs and sanitize                                             Register_Using_Email_Activity.kt 62
name = sanitizeName(binding.nameEt.text.toString().trim())                  Edit_Profile_Activity.kt 82
// Sanitize and filter the message                                          Chat_Room_Activity.kt 120
currentPassword = sanitizePassword(binding.currentPasswordEt.text.          Change_Password_Activity.kt 49
// Sanitize inputs                                                          Admin_Notices_Activity.kt 110
name = sanitizeName(binding.nameEt.text.toString().trim())                  Register_Using_Email_Activity.kt 63
email = sanitizeEmail(binding.emailEt.text.toString().trim())               Edit_Profile_Activity.kt 83
email = sanitizeEmail(binding.emailEt.text.toString().trim())               Register_Using_Email_Activity.kt 64
email = sanitizeEmail(email)                                                Log_In_Using_Email_Activity.kt 85
val sanitizedMessage = sanitizeMessage(message)                             Chat_Room_Activity.kt 121
newPassword = sanitizePassword(binding.newPasswordEt.text.toStri            Change_Password_Activity.kt 50
noticeTitle = sanitizeInput(noticeTitle)                                    Admin_Notices_Activity.kt 111
confirmNewPassword = sanitizePassword(binding.newConfirmPassw               Change_Password_Activity.kt 51
noticeDescription = sanitizeInput(noticeDescription)                        Admin_Notices_Activity.kt 112
sanitizeDob(String) in com.....reflectionsoffaith.Account.Edit_Profile_Activity   ReflectionsOfFaith.app....
sanitizeEmail(String) in com.....StartPage.Register_Using_Email_Activity   ReflectionsOfFaith.app.main
sanitizeEmail(String) in com.....reflectionsoffaith.Account.Edit_Profile_Activity   ReflectionsOfFaith.ap...
sanitizeEmail(String) in com.....StartPage.Log_In_Using_Email_Activity     ReflectionsOfFaith.app.main
sanitizeInput(String) in com.....Notices_Page.activities.Admin_Notices_Activity   ReflectionsOfFaith.ap...
sanitizeMessage(String) in com.....ChatRoom.activities.Chat_Room_Activity   ReflectionsOfFaith.app....
sanitizeName(String) in com.....StartPage.Register_Using_Email_Activity    ReflectionsOfFaith.app.main
sanitizeName(String) in com.....reflectionsoffaith.Account.Edit_Profile_Activity   ReflectionsOfFaith.ap...
sanitizePassword(String) in com.....StartPage.Register_Using_Email_Activity   ReflectionsOfFaith.app....
sanitizePassword(String) in com.....Account.Change_Password_Activity       ReflectionsOfFaith.app.main
sanitizePassword(String) in com.....StartPage.Log_In_Using_Email_Activity  ReflectionsOfFaith.app.m...
sanitizePhoneNumber(String) in com.....Account.Edit_Profile_Activity       ReflectionsOfFaith.app.main
// Method to sanitize passwords (for demonstration, could enhance f        Change_Password_Activity.kt 111
private fun sanitizePassword(input: String): String {                       Change_Password_Activity.kt 112
private fun sanitizeInput(input: String): String {                          Admin_Notices_Activity.kt 142
password = sanitizePassword(password)                                       Log_In_Using_Email_Activity.kt 86
val sanitizedMessage = sanitizeMessage(message)                             Chat_Room_Activity.kt 121
val sanitized = input.replace(Regex("<[^>]*>"), "") // Remove HTML tags     Admin_Notices_Activity.kt 144
dob = sanitizeDob(binding.dobEt.text.toString().trim())                     Edit_Profile_Activity.kt 84
```

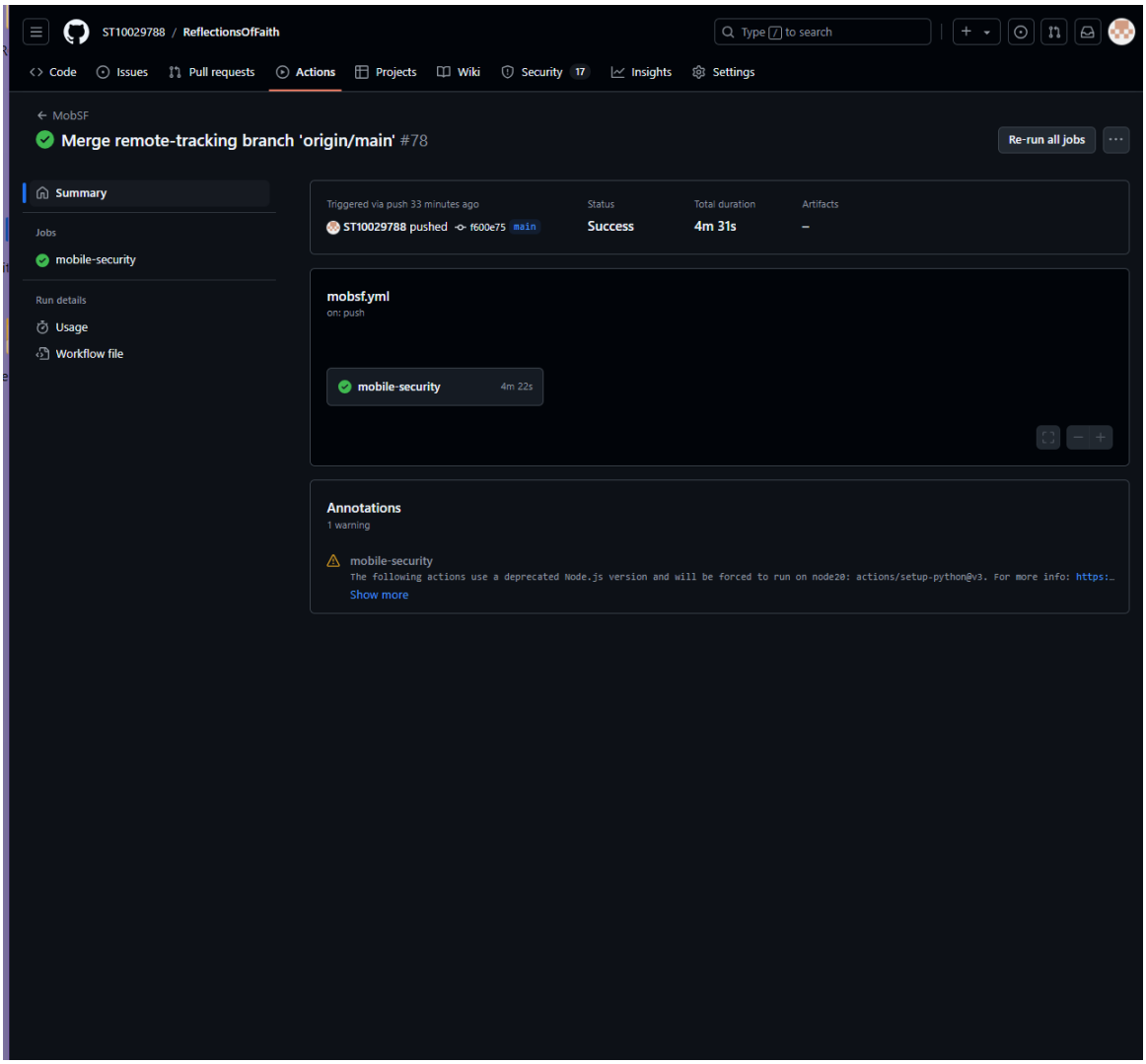
```
// Method to validate the user's input data
private fun validateData() {
    // Get the user's email and password from the input fields
    email = binding.emailEt.text.toString().trim()
    password = binding.passwordEt.text.toString().trim()

    // Log the user's email for debugging purposes
    Log.d(TAG, "validate email: $email")

    // Check if the email is valid
    if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        binding.emailEt.error = "Invalid Email Format"
        binding.emailEt.requestFocus()
    }
    // Check for password strength (length and character checks)
    else if (password.length < 6) { // Minimum length check, adjust as nee
        binding.passwordEt.error = "Password must be at least 6 characters
        binding.passwordEt.requestFocus()
    } else {
        // If the data is valid, sanitize inputs before login
        email = sanitizeEmail(email)
        password = sanitizePassword(password)

        // If inputs are sanitized, proceed to login
        loginUser()
    }
}

// Method to sanitize email
private fun sanitizeEmail(input: String): String {
    // Replace unwanted characters and return
    return input.replace("[^\\w@.-]".toRegex(), "")
}

// Method to sanitize password (you can add more rules if needed)
private fun sanitizePassword(input: String): String {
    // Replace unwanted characters (if necessary)
    return input.replace("[^\\w@!#$%^&*()-=_+<>?,.;:']".toRegex(), "")
}
```

Security

- Security tools used: SonarQube, **MobSF** and **Snyk**. Closed 86 issues, 11 issues open.


Code Analysis
Use of Hardcoded Credentials
650
Use of Hardcoded Credentials
650


SonarQube Cloud
My Projects
My Issues
Explore
ReflectionsOfFaith
PUBLIC
ST10029788 > ReflectionsOfFaith > main
Summary
Issues
Security Hotspots
Measures
Code
Activity
Overview
Main Branch
Pull Requests
Branches
Information
Administration
Collapse
10.0% Security Hotspots Reviewed
To review
Fixed
Safe
9 Security Hotspots to review
Review priority: Medium
Weak Cryptography
Make sure that using this pseudorandom number generator is safe here.
Make sure that using this pseudorandom number generator is safe here.
Make sure that using this pseudorandom number generator is safe here.
Review priority: Low
Encryption of Sensitive Data
Others
9 of 9 shown
Make sure that using this pseudorandom number generator is safe here.
Using pseudorandom number generators (PRNGs) is security-sensitive kotlin:S2245
Status: To Review
This Security Hotspot needs to be reviewed to assess whether the code poses a risk.
Review
Where is the risk?
What's the risk?
Assess the risk
How can I fix it?
Activity
Review priority:
Medium
Category:
Weak Cryptography
Assignee:
ST10029788
app/.../Donation_Market/adapter/AdapterCategory.kt
Show 35 more lines
// get Data from modelCategory
val icon = modelCategory.icon
val category = modelCategory.category
// get random color to set as the background color of the categoryIconIv
val random = Random()
Make sure that using this pseudorandom number generator is safe here.
val color = Color.argb(255, random.nextInt(255), random.nextInt(255), random.nextInt(255))
//set data to UI View of row category.xml
holder.categoryIconIv.setImageResource(icon)
holder.categoryIv.text = category
Show 25 more lines


ST10029788 / ReflectionsOfFaith
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security 17
Insights
Settings
MobSF
Merge remote-tracking branch 'origin/main' #78
Re-run all jobs
Summary
Jobs
mobile-security
Run details
Usage
Workflow file
Triggered via push 33 minutes ago
ST10029788 pushed f600e75 main
Status
Success
Total duration
4m 31s
Artifacts
–
mobsf.yml
on: push
mobile-security
4m 22s
Annotations
1 warning
mobile-security
The following actions use a deprecated Node.js version and will be forced to run on node20: actions/setup-python@v3. For more info: https:…
Show more

Overview

Reporting

Policy

Advisories

Vulnerability alerts

Dependabot

Code scanning 13

Secret scanning 4

Code scanning

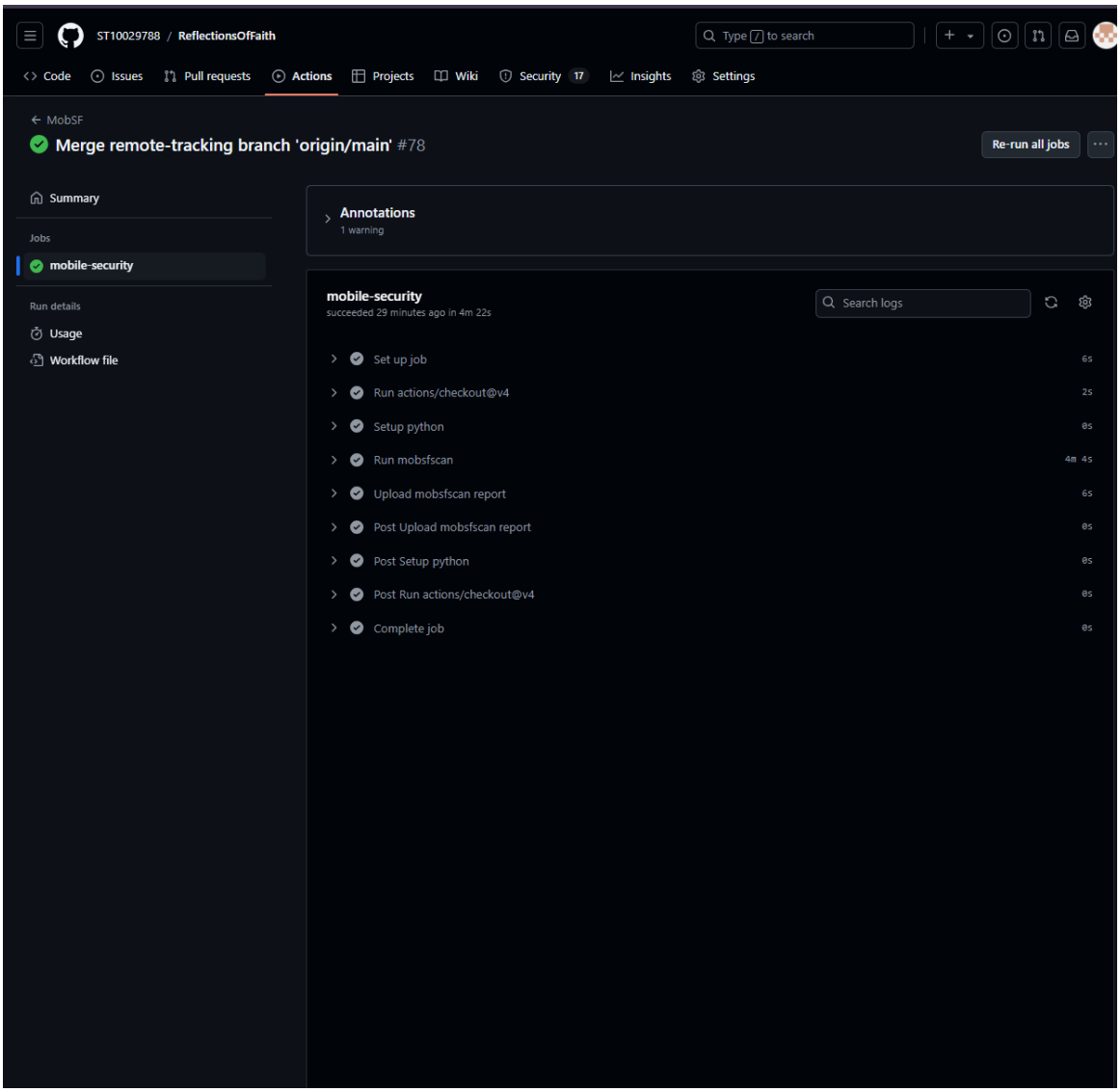
CodeQL is reporting errors. Check the status page for help.

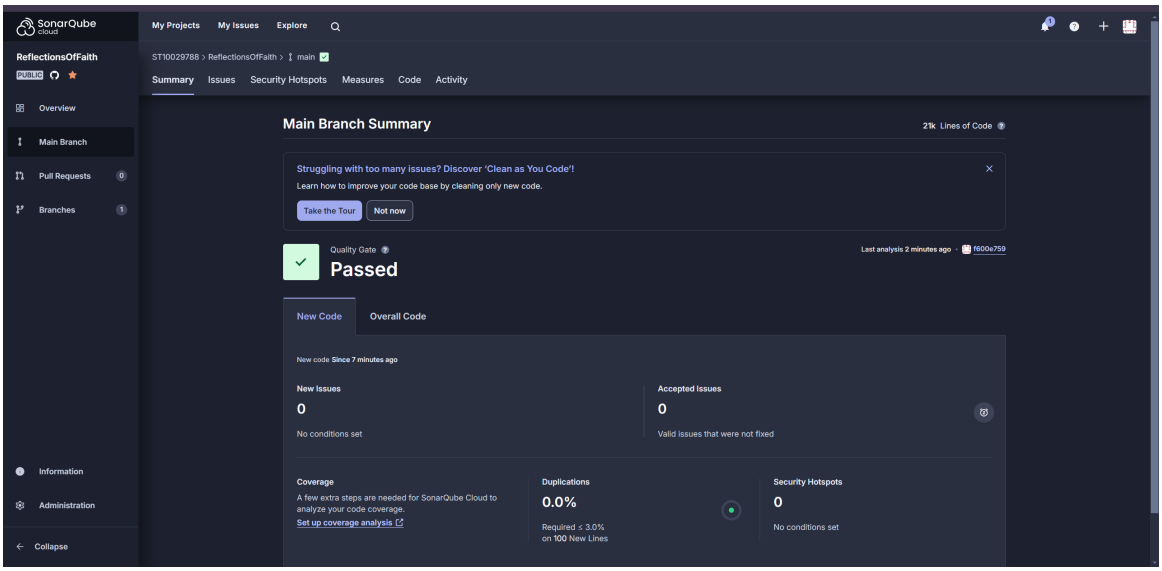
Tools 3 + Add tool

is:open branch:main

13 Open ✓ 107 Closed Language Tool Branch Rule Severity Sort

- Legacy Firebase Cloud Messaging API Key should not be disclosed High main
  #120 opened 38 minutes ago • Detected by SonarCloud in app/.../Utilities/Utils.kt :30
- Exported component access should be restricted with appropriate permissions Medium main
  #121 opened 38 minutes ago • Detected by SonarCloud in app/.../main/AndroidManifest.xml :177
- Hidden elements in view can be used to hide data from user. But this data can be leaked. Error main
  #122 opened 30 minutes ago • Detected by mobsfscan in app/.../adapter/AdapterChat.kt :77
- A hardcoded password in plain text is identified. Warning main
  #12 opened last month • Detected by mobsfscan in app/.../layout/activity_log_in_email.xm... :137
- Files may contain hardcoded sensitive information like usernames, passwords, keys etc. Warning main
  #5 opened last month • Detected by mobsfscan in app/.../fragments/ChatsFragment.kt :108
- The App uses an insecure Random Number Generator. Warning main
  #4 opened last month • Detected by mobsfscan in app/.../Utilities/Firebase_Cloud_Messaging... :16
- The App logs information. Sensitive information should never be logged. Note main
  #25 opened last month • Detected by mobsfscan in app/.../adapter/AdapterChat.kt :87
- This app does not use TLS/SSL certificate or public key pinning to detect or prevent MITM attacks in secure communication channel. Note main
  #11 opened last month • Detected by mobsfscan in . :1
- This app does not uses SafetyNet Attestation API that provides cryptographically-signed attestation, assessing the device's integrity. This check helps to ensure that the servers are interacting with the genuine app running on a genuine Android device. Note main
  #10 opened last month • Detected by mobsfscan in . :1
- This app does not have capabilities to prevent tapjacking attacks. Note main
  #9 opened last month • Detected by mobsfscan in . :1
- This app does not have root detection capabilities. Running a sensitive application on a rooted device questions the device integrity and affects users data. Note main
  #8 opened last month • Detected by mobsfscan in . :1
- This app does not have capabilities to prevent against Screenshots from Recent Task History/ Now On Tap etc. Note main
  #7 opened last month • Detected by mobsfscan in . :1
- This app does not enforce TLS Certificate Transparency which helps to detect SSL certificates that have been mistakenly issued by a certificate authority or maliciously acquired from an otherwise unimpeachable certificate authority. Note main
  #6 opened last month • Detected by mobsfscan in . :1


ST10029788 / ReflectionsOfFaith
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security 17
Insights
Settings
MobSF
Merge remote-tracking branch 'origin/main' #78
Re-run all jobs
Summary
Jobs
mobile-security
Run details
Usage
Workflow file
Annotations
1 warning
mobile-security
succeeded 29 minutes ago in 4m 22s
Search logs
Set up job 6s
Run actions/checkout@v4 2s
Setup python 0s
Run mobsfscan 4m 4s
Upload mobsfscan report 6s
Post Upload mobsfscan report 0s
Post Setup python 0s
Post Run actions/checkout@v4 0s
Complete job 0s


SonarQube cloud
My Projects
My Issues
Explore
ReflectionsOfFaith
PUBLIC
ST10029788 > ReflectionsOfFaith > main
Summary
Issues
Security Hotspots
Measures
Code
Activity
Overview
Main Branch
Pull Requests 0
Branches 1
Information
Administration
Collapse
Main Branch Summary
21k Lines of Code
Struggling with too many issues? Discover 'Clean as You Code'!
Learn how to improve your code base by cleaning only new code.
Take the Tour
Not now
Quality Gate
Passed
Last analysis 2 minutes ago
f900e759
New Code
Overall Code
New code Since 7 minutes ago
New Issues
0
No conditions set
Accepted Issues
0
Valid issues that were not fixed
Coverage
A few extra steps are needed for SonarQube Cloud to analyze your code coverage.
Set up coverage analysis
Duplications
0.0%
Required ≤ 3.0% on 100 New Lines
Security Hotspots
0
No conditions set