HashHawk

Redacted

Redacted

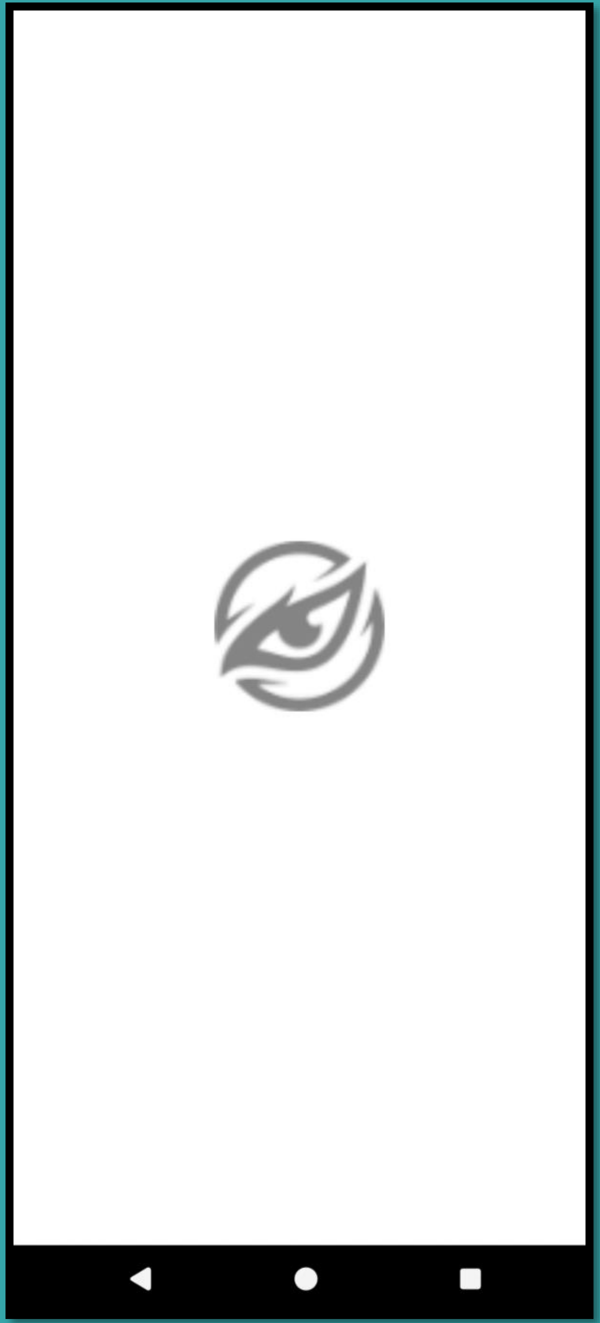
Scenario outlining purpose

- Imagine if you have a malware sample on a machine disconnected from the internet
- You can hash the file on the local machine using software
- Can’t use virustotal to check it as that’s online
- Can’t copy hash from PC to host PC as allowing shared clipboard is an explicit infection vector
- How do you check this without manually typing it out?

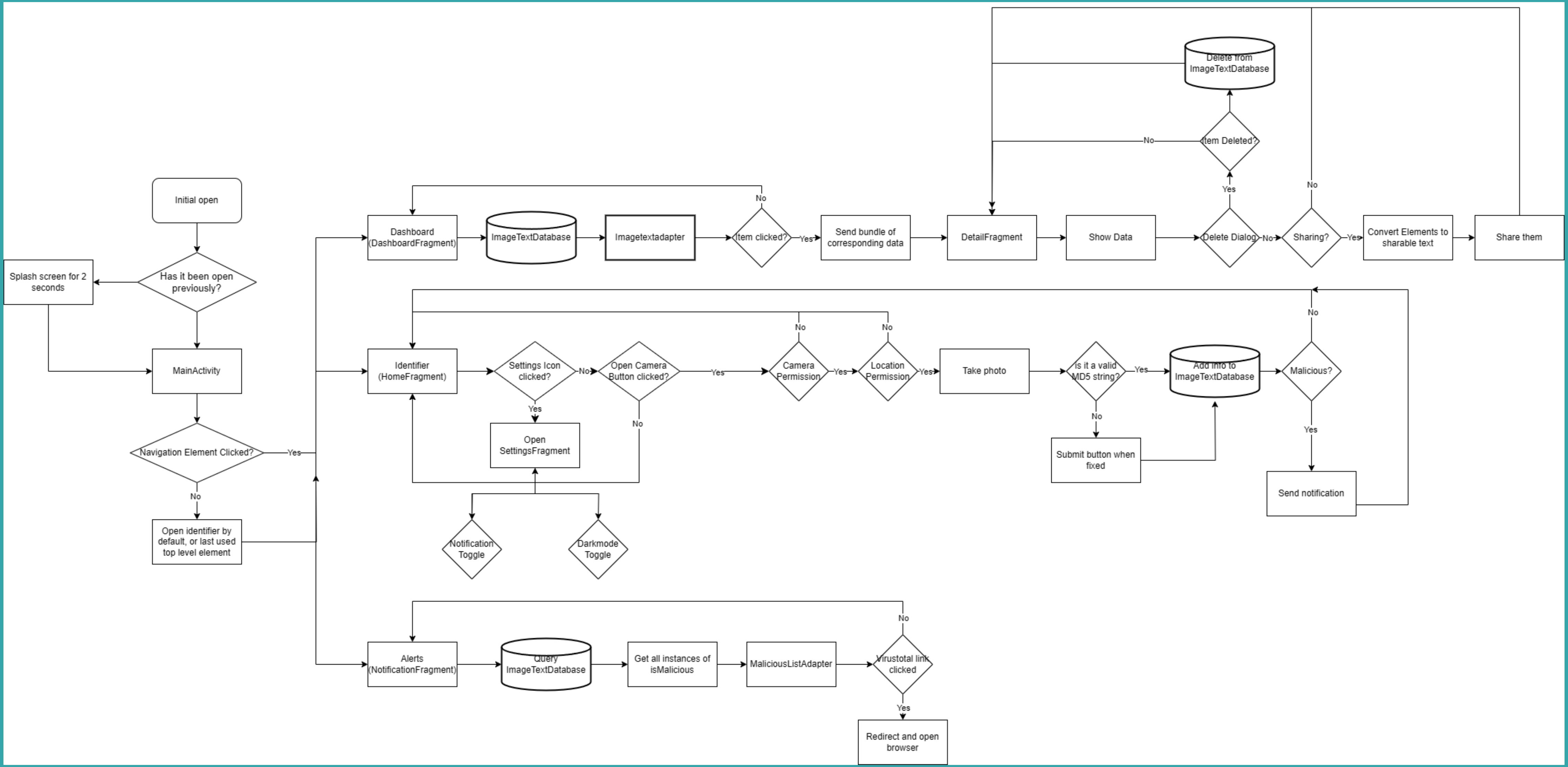
HashHawk

- Intended to identify MD5 hashes from a taken image, log relevant information and automatically identify if they are malicious
- OCR (optical character recognition) using Firebase ML kit
- Elements such as date, MD5 and location stored in a database
- Querying VirusTotal API to check if elements are malicious

DEMONSTRATION



App Flow Diagram

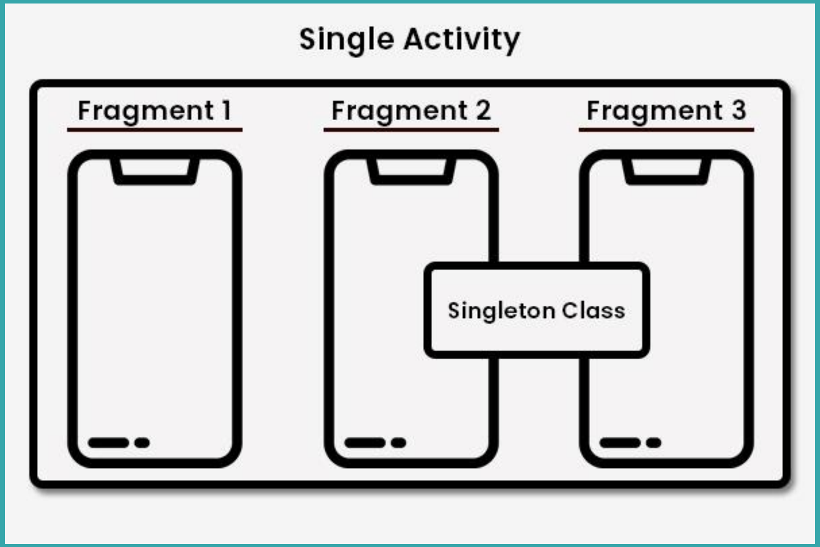

Initial open
Has it been open previously?
Splash screen for 2 seconds
MainActivity
Navigation Element Clicked?
Yes
No
Open identifier by default, or last used top level element
Dashboard (DashboardFragment)
ImageTextDatabase
Imagetextadapter
Item clicked?
No
Yes
Send bundle of corresponding data
DetailFragment
Show Data
Delete Dialog
Yes
Item Deleted?
No
Delete from ImageTextDatabase
Sharing?
No
Yes
Convert Elements to sharable text
Share them
Identifier (HomeFragment)
Settings Icon clicked?
No
Yes
Open SettingsFragment
Notification Toggle
Darkmode Toggle
Open Camera Button clicked?
No
Yes
Camera Permission
No
Yes
Location Permission
No
Yes
Take photo
Is it a valid MD5 string?
Yes
No
Submit button when fixed
Add info to ImageTextDatabase
Malicious?
No
Yes
Send notification
Alerts (NotificationFragment)
Query ImageTextDatabase
Get all instances of isMalicious
MaliciousListAdapter
Virustotal link clicked
No
Yes
Redirect and open browser

General Architecture

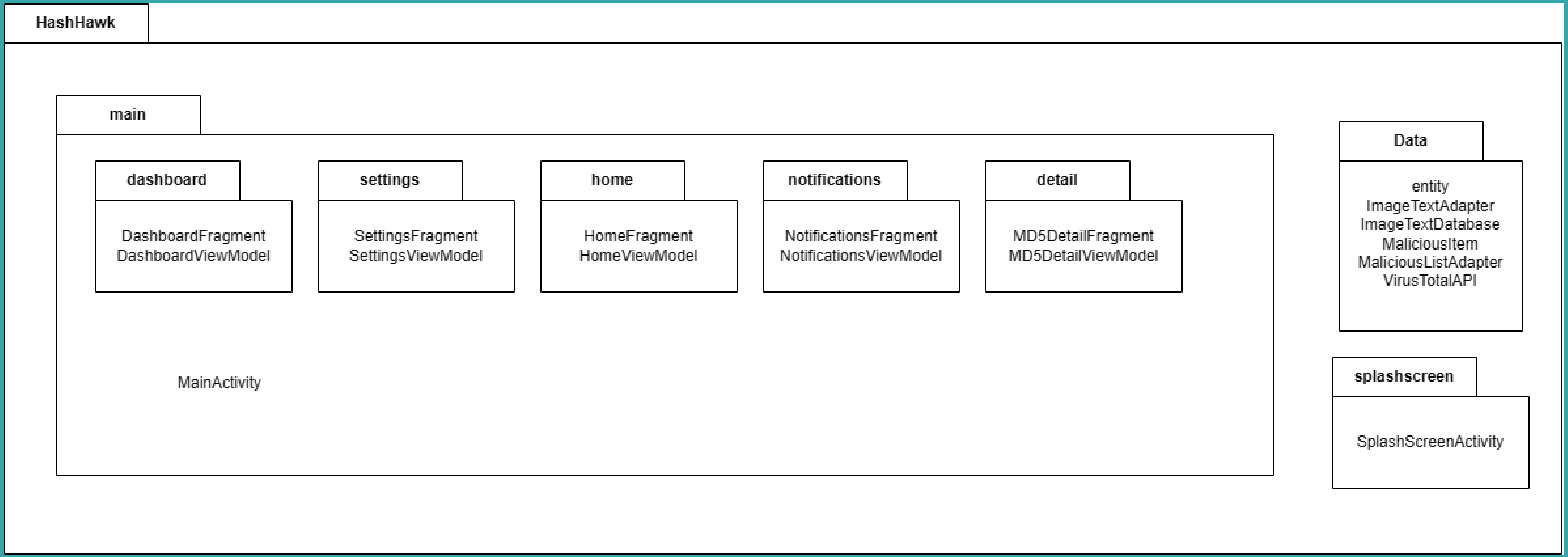
- Kotlin used owing to it being a modern language for native android development
- Local storage made use of room persistence as it was deemed preferable to SQL.
- Bundle to transfer data between fragments

Justification of Structure

- Single activity flow (sans splash)
- Navigation based on fragments
- MVVM model
- Optimized memory usage
- Dynamic UI
- Ease of resource sharing



Class Relation Diagram

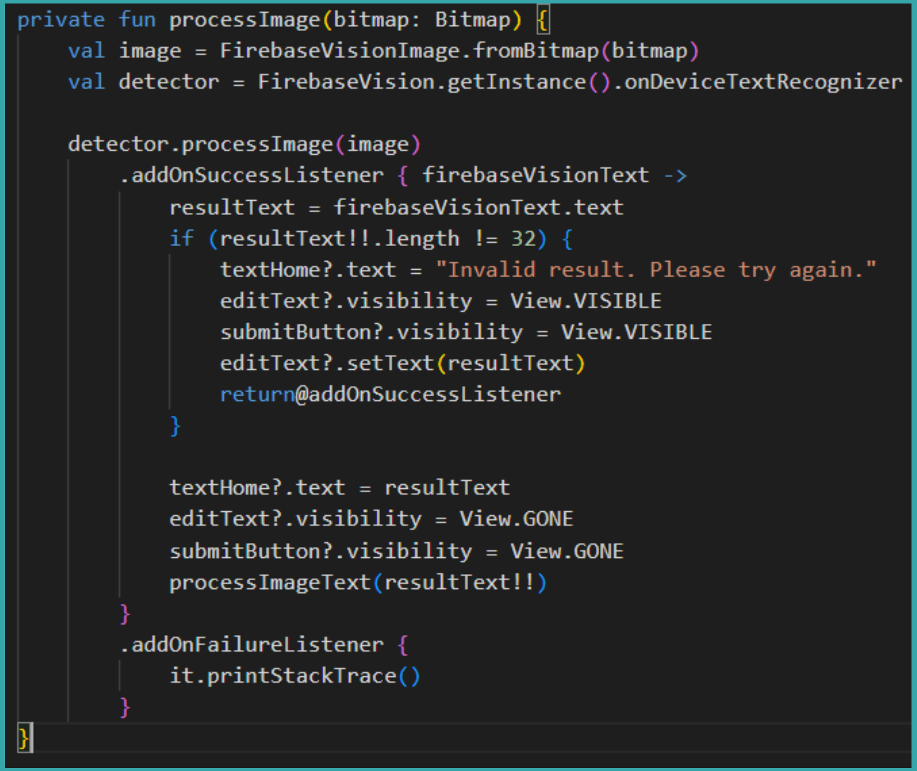


Main features

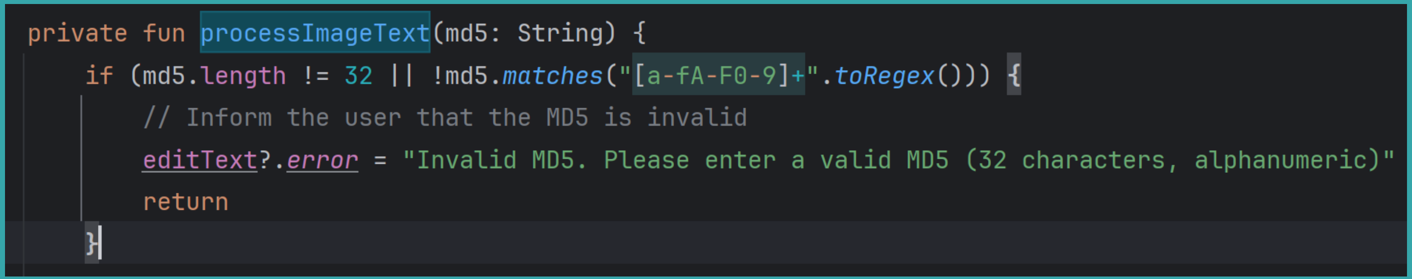
- Text recognition
- Database storage for data
- Splash screen
- Notifications when malicious elements found
- Virustotal API queries and response
- Dark/Light mode via shared preferences

Text Recognition

- Use Firebase vision from image
- Run check against obtained text, see if it’s 32 characters (MD5 length)
- If not, make correction elements visible.
- Make sure it’s only numbers or letters



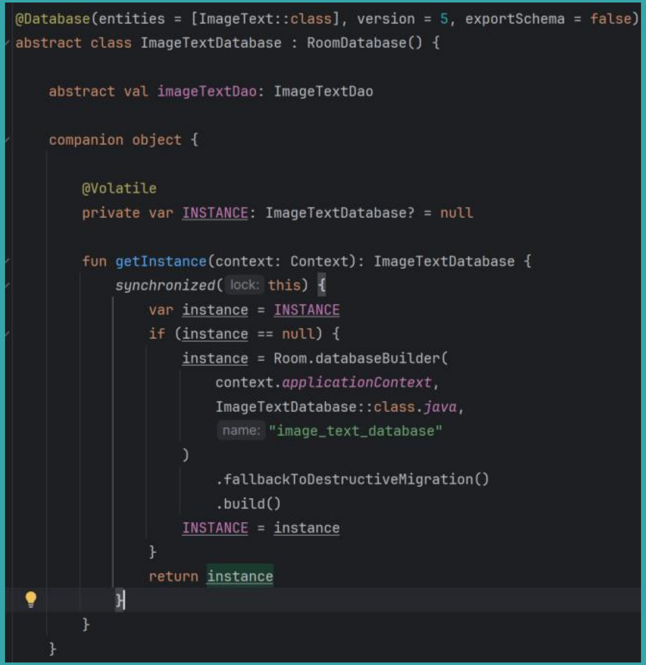
```kotlin
private fun processImage(bitmap: Bitmap) {
    val image = FirebaseVisionImage.fromBitmap(bitmap)
    val detector = FirebaseVision.getInstance().onDeviceTextRecognizer

    detector.processImage(image)
        .addOnSuccessListener { firebaseVisionText ->
            resultText = firebaseVisionText.text
            if (resultText!!.length != 32) {
                textHome?.text = "Invalid result. Please try again."
                editText?.visibility = View.VISIBLE
                submitButton?.visibility = View.VISIBLE
                editText?.setText(resultText)
                return@addOnSuccessListener
            }

            textHome?.text = resultText
            editText?.visibility = View.GONE
            submitButton?.visibility = View.GONE
            processImageText(resultText!!)
        }
        .addOnFailureListener {
            it.printStackTrace()
        }
}
```



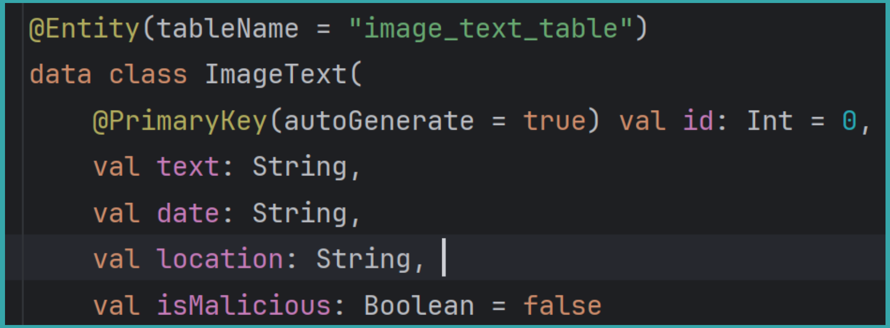
```kotlin
private fun processImageText(md5: String) {
    if (md5.length != 32 || !md5.matches("[a-fA-F0-9]+".toRegex())) {
        // Inform the user that the MD5 is invalid
        editText?.error = "Invalid MD5. Please enter a valid MD5 (32 characters, alphanumeric)"
        return
    }
```

Room database

- Entity to setup what a given database element has
- Room database built and uses multithreading
- @Voltatile ensures threading runs smoothly
- Synchronized protects from concurrent execution by multiple threads



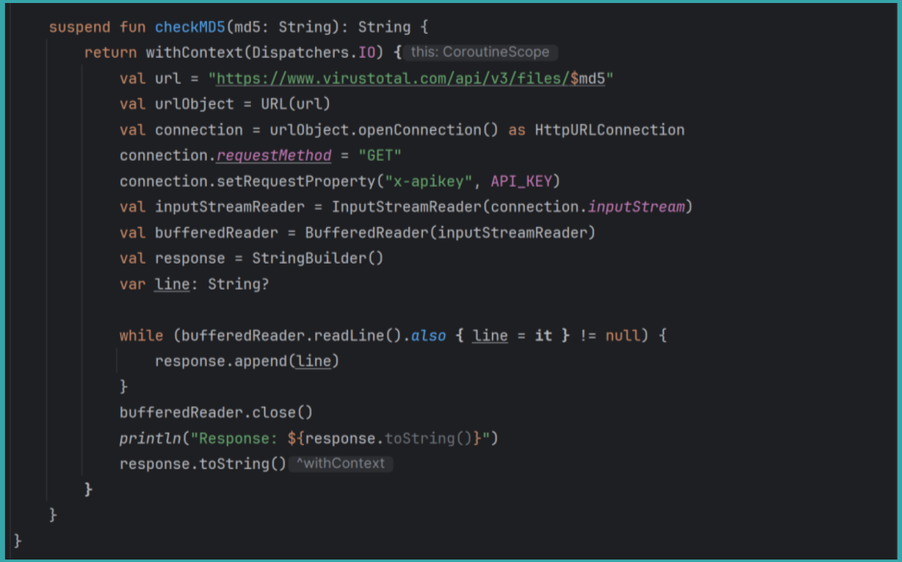
```kotlin
@Database(entities = [ImageText::class], version = 5, exportSchema = false)
abstract class ImageTextDatabase : RoomDatabase() {

    abstract val imageTextDao: ImageTextDao

    companion object {

        @Volatile
        private var INSTANCE: ImageTextDatabase? = null

        fun getInstance(context: Context): ImageTextDatabase {
            synchronized( lock: this) {
                var instance = INSTANCE
                if (instance == null) {
                    instance = Room.databaseBuilder(
                        context.applicationContext,
                        ImageTextDatabase::class.java,
                         name: "image_text_database"
                    )
                        .fallbackToDestructiveMigration()
                        .build()
                    INSTANCE = instance
                }
                return instance
            }
        }
    }
```



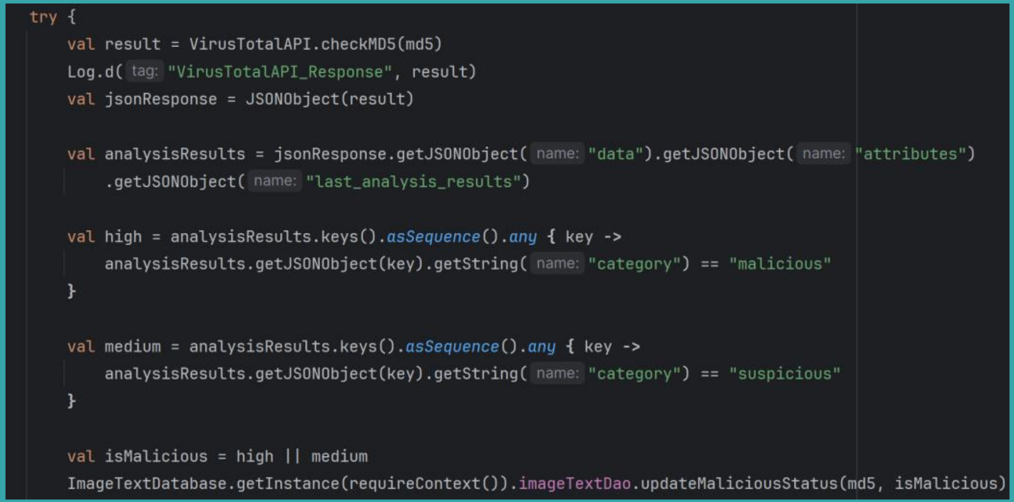
```kotlin
@Entity(tableName = "image_text_table")
data class ImageText(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val text: String,
    val date: String,
    val location: String,
    val isMalicious: Boolean = false
```

API querying

- HTTP communication with Virustotal API – setup an object
- Using Virustotal's free API
- Returns a JSON
- Data is checked for Medium/High vulns (Malicious & Suspicious)
- Updates isMalicious in database



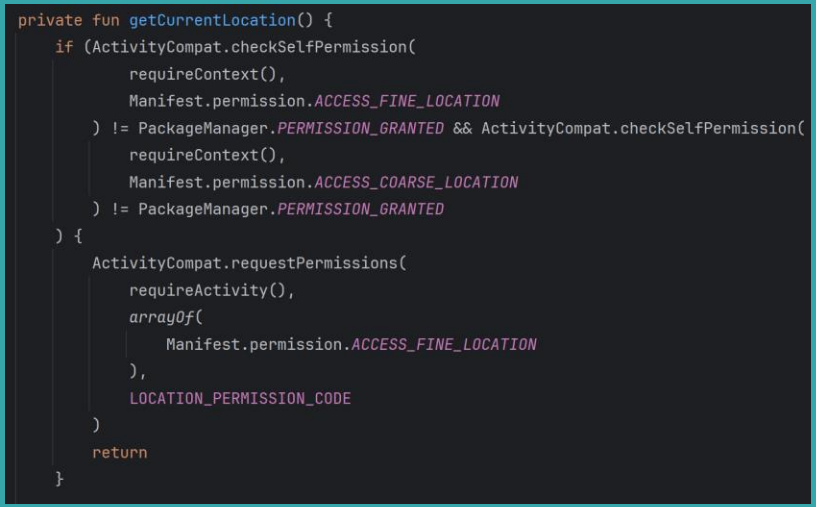
```kotlin
    suspend fun checkMD5(md5: String): String {
        return withContext(Dispatchers.IO) { this: CoroutineScope
            val url = "https://www.virustotal.com/api/v3/files/$md5"
            val urlObject = URL(url)
            val connection = urlObject.openConnection() as HttpURLConnection
            connection.requestMethod = "GET"
            connection.setRequestProperty("x-apikey", API_KEY)
            val inputStreamReader = InputStreamReader(connection.inputStream)
            val bufferedReader = BufferedReader(inputStreamReader)
            val response = StringBuilder()
            var line: String?

            while (bufferedReader.readLine().also { line = it } != null) {
                response.append(line)
            }
            bufferedReader.close()
            println("Response: ${response.toString()}")
            response.toString() ^withContext
        }
    }
}
```



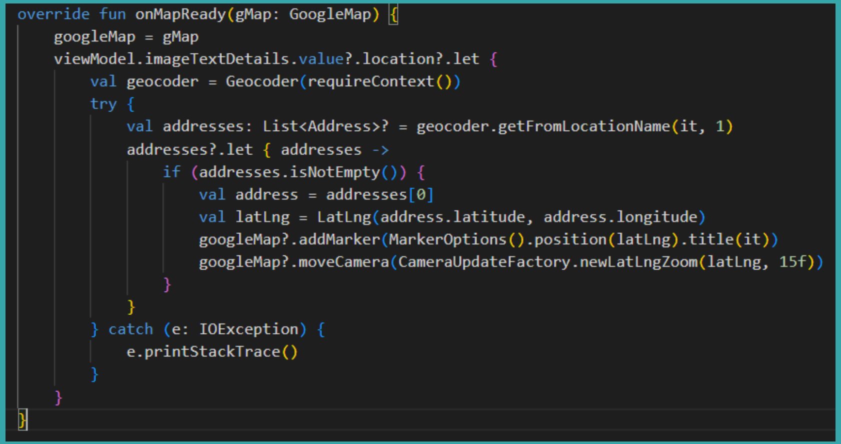
```kotlin
try {
    val result = VirusTotalAPI.checkMD5(md5)
    Log.d( tag: "VirusTotalAPI_Response", result)
    val jsonResponse = JSONObject(result)

    val analysisResults = jsonResponse.getJSONObject( name: "data").getJSONObject( name: "attributes")
        .getJSONObject( name: "last_analysis_results")

    val high = analysisResults.keys().asSequence().any { key ->
        analysisResults.getJSONObject(key).getString( name: "category") == "malicious"
    }

    val medium = analysisResults.keys().asSequence().any { key ->
        analysisResults.getJSONObject(key).getString( name: "category") == "suspicious"
    }

    val isMalicious = high || medium
    ImageTextDatabase.getInstance(requireContext()).imageTextDao.updateMaliciousStatus(md5, isMalicious)
```

Location

- Use Android to get users location (Coarse, Fine or Last)
- Geocoding used to convert location to a visual address with marker



```kotlin
private fun getCurrentLocation() {
    if (ActivityCompat.checkSelfPermission(
            requireContext(),
            Manifest.permission.ACCESS_FINE_LOCATION
        ) != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(
            requireContext(),
            Manifest.permission.ACCESS_COARSE_LOCATION
        ) != PackageManager.PERMISSION_GRANTED
    ) {
        ActivityCompat.requestPermissions(
            requireActivity(),
            arrayOf(
                Manifest.permission.ACCESS_FINE_LOCATION
            ),
            LOCATION_PERMISSION_CODE
        )
        return
    }
```



```kotlin
override fun onMapReady(gMap: GoogleMap) {
    googleMap = gMap
    viewModel.imageTextDetails.value?.location?.let {
        val geocoder = Geocoder(requireContext())
        try {
            val addresses: List<Address>? = geocoder.getFromLocationName(it, 1)
            addresses?.let { addresses ->
                if (addresses.isNotEmpty()) {
                    val address = addresses[0]
                    val latLng = LatLng(address.latitude, address.longitude)
                    googleMap?.addMarker(MarkerOptions().position(latLng).title(it))
                    googleMap?.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, 15f))
                }
            }
        } catch (e: IOException) {
            e.printStackTrace()
        }
    }
}
```

Overview and Critical Analysis

Overall Analysis

- OCR with firebase is only moderately accurate
- Still faster than manually
- Alert feature useful as it informs the user of MD5 state quickly
- Core functionality successful
- Large scope for expansion

Usability

Positives

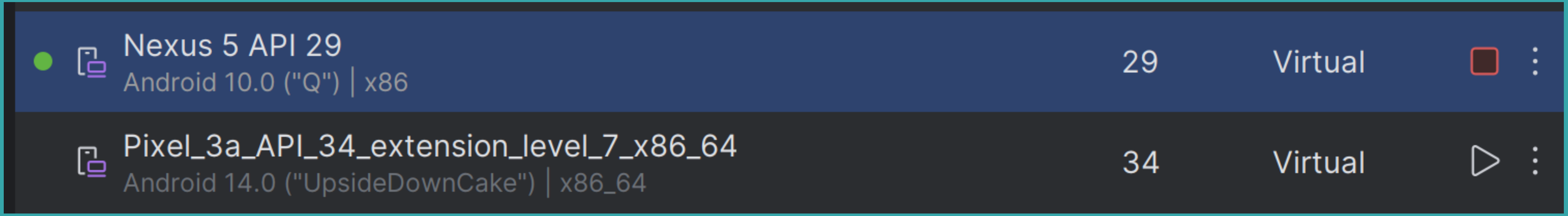
- Light and dark mode following system theme
- Handling of errors such as OCR failures
- Some labeling in place for screen readers

Negatives

- Design could be improved, functional – but not overtly attractive. Utilitarian.
- Colour blindness themes for accessibility

Compatibility

- Tested on both the Nexus 5 recommended test phone
- Tested on current android Sony Xperia 1 V
- Rotation functionality and parent resizing indicates likely compatibility with larger devices (tablets)



Nexus 5 API 29 Android 10.0 ("Q") | x86	29	Virtual	
Pixel_3a_API_34_extension_level_7_x86_64 Android 14.0 ("UpsideDownCake") | x86_64	34	Virtual	

Performance considerations

- Network retrieval only occurs once per element lifespan
- Battery life unlikely to be a major consideration owing to the lack of persistently running services
- Storage was explicitly considered
- Recyclerview used to display large datasets better than Listview
- Previously mentioned single activity multiple fragment format

Security

Permissions

- Camera, for OCR
- Network , For VirusTotal
- Notifications, but this is toggleable
- Location, both fine and coarse.

Other elements

- Room is used for the majority of storage
- Shared preferences provides no noteworthy information
- Hardcoded API keys

Future Features

- Image upload from gallery as opposed to having to be taken live.
- Without requirement for usage of ONLY android and google APIs, replacing imageview with photoview.
- Splash screen modification depending on dark/lightmode.

Thank you for listening

References

- Android Developers, 2023. Camera support. [Online] Available at: https://developer.android.com/studio/run/emulator-use-camera [Accessed 10 May 2024].
- Android Developers, 2024. Communicate with fragments. [Online] Available at: https://developer.android.com/guide/fragments/communicate [Accessed 12 May 2024].
- Android Developers, 2024. Fragments. [Online] Available at: https://developer.android.com/guide/navigation/design#fragments [Accessed 14 May 2024].
- Android Developers, 2024. Navigation and the back stack. [Online] Available at: https://developer.android.com/guide/navigation/backstack [Accessed 14 May 2024].
- Android Developers, 2024. R.id. [Online] Available at: https://developer.android.com/reference/android/R.id [Accessed 11 April 2024].
- Android Developers, 2024. Save data in a local database using Room. [Online] Available at: https://developer.android.com/training/data-storage/room [Accessed 10 May 2024].
- Android Developers, 2024. Adapter. [Online] Available at: https://developer.android.com/reference/android/widget/Adapter [Accessed 14 May 2024].

References

- Esracangungor, 2023. Handling Screen Orientation Changes in Android with Kotlin. [Online] Available at: https://medium.com/@esracangungor/handling-screen-orientation-changes-in-android-with-kotlin-1c20713f986b [Accessed 13 May 2024].
- Google Maps Platform, 2024. Maps SDK for Android Quickstart. [Online] Available at: https://developers.google.com/maps/documentation/android-sdk/start#api-key [Accessed 14 May 2024].
- Kocovic, Z., 2023. Pass data between fragments in Kotlin. [Online] Available at: https://medium.com/@kocoviczoran_5004/pass-data-between-fragments-in-kotlin-6928c08d2bd0 [Accessed 14 May 2024].
- Mbano, U., 2022. Android Room versus SQLite — Which is best?. [Online] Available at: https://medium.com/dvt-engineering/android-room-versus-sqlite-which-is-best-32ff651bc361 [Accessed 12 May 2024].
- Necanli, B., 2023. Single Activity vs Multiple Activities Architecture. [Online] Available at: https://medium.com/appcent/a-single-activity-vs-multiple-activities-architecture-96a23b783036 [Accessed 11 May 2024].
- Shutterstock, 2019. Eagle Eyes Bird Hawk Logo Design Inspiration. [Online] Available at: https://www.shutterstock.com/image-vector/eagle-eyes-bird-hawk-logo-design-1346883548 [Accessed 10 April 2024].

References

- Stack Overflow, 2019. Answer to "Change Drawable Based on Theme". [Online] Available at: https://stackoverflow.com/questions/13735675/change-drawable-based-on-theme [Accessed 14 May 2024].
- VirusTotal, 2023. VirusTotal API v3 Overview. [Online] Available at: https://docs.virustotal.com/reference/overview [Accessed 14 May 2024].