MC Assignment 3

Arnav Agarwal
2021235

Question 1: Matrix Calculator

MainActivity.kt

- **Supports input of two matrices** with customizable dimensions
- Performs addition, subtraction, multiplication, and division using **native C++** methods via JNI
- **Validates dimensions** and compatibility before operations

```
38
39      private fun addMatrixInputs(label: String, rows: Int, cols: Int, list: MutableList<EditText>) {
40          val title = TextView( context: this).apply {
41              text = label
42              textSize = 16f
43              setPadding( left: 0,  top: 16,  right: 0,  bottom: 8)
44          }
45          matrixInputsContainer.addView(title)
46
47          for (i in 0 ≤ until < rows) {
48              val rowLayout = LinearLayout( context: this).apply {
49                  orientation = LinearLayout.HORIZONTAL
50              }
51
52              for (j in 0 ≤ until < cols) {
53                  val cell = EditText( context: this).apply {
54                      layoutParams = LinearLayout.LayoutParams( width: 0, ViewGroup.LayoutParams.WRAP_CONTENT,
55                          marginEnd = 4
56                      }
57                      inputType = InputType.TYPE_CLASS_NUMBER or InputType.TYPE_NUMBER_FLAG_DECIMAL or InputTy
58                      gravity = Gravity.CENTER
59                      setPadding( left: 8,  top: 4,  right: 8,  bottom: 4)
60                      background = ContextCompat.getDrawable(context, android.R.drawable.editbox_background)
61                  }
62                  list.add(cell)
63                  rowLayout.addView(cell)
64              }
65              matrixInputsContainer.addView(rowLayout)
```

- Dynamically generates matrix input fields using `EditText` and `LinearLayout`
- Displays results in a neatly formatted matrix view
- Loads native library `matrixops` on startup

```
val result: FloatArray = try {
    when (operation) {
        "add" -> {
            if (rowsA != rowsB || colsA != colsB) {
                showToast( msg: "Add/Subtract requires equal matrix dimensions")
                return@setOnClickListener
            }
            addMatrices(matrixA, matrixB, rowsA, colsA)
        }

        "subtract" -> {
            if (rowsA != rowsB || colsA != colsB) {
                showToast( msg: "Add/Subtract requires equal matrix dimensions")
                return@setOnClickListener
            }
            subtractMatrices(matrixA, matrixB, rowsA, colsA)
        }

        "multiply" -> {
            if (colsA != rowsB) {
                showToast( msg: "Multiplication requires cols(A) == rows(B)")
                return@setOnClickListener
            }
            multiplyMatrices(matrixA, matrixB, rowsA, colsA, colsB)
        }
```

- Shows **error messages** using `Toast` for **invalid inputs or mismatched dimensions.**
- Handles matrix input reading and formatting with helper functions

Matrixops.cpp

```
extern "C"
JNIEXPORT jfloatArray JNICALL
Java_com_example_matrixcalculator_MainActivity_addMatrices(
        JNIEnv *env, jobject MainActivity , jfloatArray matA, jfloatArray matB, jint rows, jint cols) {

    int size = rows * cols;
    jfloat* a = env->GetFloatArrayElements( array: matA,  isCopy: nullptr);
    jfloat* b = env->GetFloatArrayElements( array: matB,  isCopy: nullptr);

    auto A : Matrix<float, -1, -1, 1>  = toMatrix( array: a, rows, cols);
    auto B : Matrix<float, -1, -1, 1>  = toMatrix( array: b, rows, cols);
    Matrix<float, Dynamic, Dynamic, RowMajor> C = A + B;

    jfloatArray result = env->NewFloatArray( length: size);
    env->SetFloatArrayRegion( array: result,  start: 0,  len: size,  buf: C.data());

    env->ReleaseFloatArrayElements( array: matA,  elems: a,  mode: JNI_ABORT);
    env->ReleaseFloatArrayElements( array: matB,  elems: b,  mode: JNI_ABORT);

    return result;
}
```

- Implements native methods for matrix operations: **addition, subtraction, multiplication, and division**
- Uses Eigen's `Matrix<float, Dynamic, Dynamic, RowMajor>` for efficient matrix computations

- **Accepts matrices from Java (`jfloatArray`), maps them to Eigen matrices using `Map`**
- Performs **element-wise operations for addition and subtraction**

```
extern "C"
JNIEXPORT jfloatArray JNICALL
Java_com_example_matrixcalculator_MainActivity_multiplyMatrices(
        JNIEnv *env, jobject MainActivity , jfloatArray matA, jfloatArray matB,
        jint rowsA, jint colsA, jint colsB) {

    jfloat* a = env->GetFloatArrayElements( array: matA,  isCopy: nullptr);
    jfloat* b = env->GetFloatArrayElements( array: matB,  isCopy: nullptr);

    auto A : Matrix<float, -1, -1, 1>  = toMatrix( array: a,  rows: rowsA,  cols: colsA);
    auto B : Matrix<float, -1, -1, 1>  = toMatrix( array: b,  rows: colsA,  cols: colsB);  // colsA == rowsB
    Matrix<float, Dynamic, Dynamic, RowMajor> C = A * B;

    jfloatArray result = env->NewFloatArray( length: rowsA * colsB);
    env->SetFloatArrayRegion( array: result,  start: 0,  len: rowsA * colsB,  buf: C.data());

    env->ReleaseFloatArrayElements( array: matA,  elems: a,  mode: JNI_ABORT);
    env->ReleaseFloatArrayElements( array: matB,  elems: b,  mode: JNI_ABORT);

    return result;
}
```

- Performs **matrix multiplication using standard `A * B`**
- For division, **multiplies A with inverse of B (only if B is square and invertible)**
- Returns the result as a `jfloatArray` back to the Android/Kotlin layer
- Handles memory safely with JNI functions like `GetFloatArrayElements`, `ReleaseFloatArrayElements`, and `SetFloatArrayRegion`
- Includes **basic error handling for non-invertible matrices in division (returns zero matrix)**

Activity_main.xml

- Uses a `ScrollView` with a vertical `LinearLayout` for scrollable UI
- Takes user input for **dimensions of Matrix A and Matrix B** using `EditText` fields
- Button (`generateMatricesBtn`) dynamically creates matrix input fields in `matrixInputsContainer`
- Includes a `RadioGroup` for selecting operations: Add, Subtract, Multiply, Divide
- A `Calculate` button (`calculateBtn`) performs the selected operation

- Displays result in a styled `TextView` (`resultTextView`) with label `Result:` above it
- Layout is clean, centered, and user-friendly with proper spacing and padding

Question 2: WiFi Application

MainActivity.kt

```
20      class MainActivity : AppCompatActivity() {                    ⚠6 ⚠3
117         private fun startLogging() {
132
133             isLogging = true
134
135             scanReceiver = object : BroadcastReceiver() {
136                 @SuppressLint("MissingPermission")
137                 override fun onReceive(context: Context?, intent: Intent?) {
138                     val results: List<ScanResult> = wifiManager.scanResults
139                     val strongestSignal = results.maxByOrNull { it.level }?.level
140                     Log.d( tag: "Values received", strongestSignal.toString())
141
142                     if (strongestSignal != null) {
143                         val list = dataMap[currentLocation] ?: mutableListOf()
144                         Log.d( tag: "Values received ${list.size}", list.toString())
145                         if (list.size < 100) {
146                             list.add(strongestSignal)
147                             dataMap[currentLocation] = list
148                             tvSummary.text = "Logging $currentLocation: ${list.size} / 100"
149                             scheduleNextScan()
150                         } else {
151                             Toast.makeText(
152                                 context: this@MainActivity,
153                                 text: "$currentLocation already has 100 samples",
154                                 Toast.LENGTH_SHORT
155                             ).show()
156                             stopLogging()
157                         }
158                     }
```

```kotlin
private fun getRssiDistribution(list: List<Int>): Map<String, Int> {
    val distribution = mutableMapOf(
        "📶 Excellent (-40 to -50)" to 0,
        "👍 Good (-51 to -60)" to 0,
        "😐 Fair (-61 to -70)" to 0,
        "⚠️ Weak (< -70)" to 0
    )
    for (rssi in list) {
        when {
            rssi >= -50 -> distribution["📶 Excellent (-40 to -50)"] = distribution["📶 Excellent (-40 to -50)"]!! +
            rssi in -60 ≤ .. ≤ -51 -> distribution["👍 Good (-51 to -60)"] = distribution["👍 Good (-51 to -60)"]!! +
            rssi in -70 ≤ .. ≤ -61 -> distribution["😐 Fair (-61 to -70)"] = distribution["😐 Fair (-61 to -70)"]!! +
            else -> distribution["⚠️ Weak (< -70)"] = distribution["⚠️ Weak (< -70)"]!! + 1
        }
    }
    return distribution
}
```

```kotlin
private fun saveMatrixToCSV() {
    val locA = dataMap["Location A"] ?: emptyList()
    val locB = dataMap["Location B"] ?: emptyList()
    val locC = dataMap["Location C"] ?: emptyList()

    if (locA.size < 100 || locB.size < 100 || locC.size < 100) {
        Toast.makeText( context: this,  text: "Collect 100 samples at all locations before saving.", Toast.LENGTH_LONG).sh
        return
    }

    val file = File(filesDir,  child: "rss_matrix.csv")
    try {
        file.printWriter().use { out ->
            out.println("Index,Location A,Location B,Location C")
            for (i in 0 ≤  until  < 100) {
                out.println("${i + 1},${locA[i]},${locB[i]},${locC[i]}")
            }
        }
        Toast.makeText( context: this,  text: "Saved to rss_matrix.csv", Toast.LENGTH_SHORT).show()
    } catch (e: Exception) {
        e.printStackTrace()
        Toast.makeText( context: this,  text: "Failed to save matrix", Toast.LENGTH_SHORT).show()
    }
}
```

- **Logs WiFi signal strength (RSSI)** at three predefined locations (A, B, C)
- Uses **`WifiManager` to scan and collect up to 100 samples** per location at 5-second intervals
- UI includes buttons to start/stop logging, view summary, and save to CSV
- **Stores collected RSSI values in `dataMap`** and allows CSV export (`rss_matrix.csv`)

- Provides **RSSI summary statistics:** min, max, average, standard deviation, and signal quality distribution
- **Requests and checks location permissions** and services before scanning
- **Displays user feedback via `Toast`** and updates summary in a `TextView`
- **Auto-stops logging when 100 samples** are collected or on activity destroy
- Allows location selection via `RadioGroup` with real-time status updates

Activity_main.xml

- **Vertical `LinearLayout`** with padding for clean layout
- **Location selection via `RadioGroup`** with three `RadioButton` options: next to router, center of room, next room
- Buttons to start/stop logging, save to CSV, and show summary
- Uses **`ScrollView` to display detailed summary** output in `TextView` (`tvSummary`)