期末專題報告

課程名稱:APP 設計

照片分類 APP

開發語言、工具：Android studio Kotlin 、深度學習、

影像處理、資料庫設計

S0854045 資工四 曾筠惠

民國 110 年 6 月 19 日

內容

需求分析

說明:

IG 上看到很多網美的版面都是同一個色調，例如: 冷色調、暖色調等等，版面乾淨、整齊可以上傳照片，而這些照片是怎麼從手機相簿幾千張照片找出來的呢，要如何從幾千張照片找到那一張接近版面色調的照片呢?

常見 IG 色調		
		
冷色調	暖色調	白色調

此外我發現版面看起來會如此整齊原因有三點 分別是 **1.物件 2.背景 3.調色** 可以利用同一色系物件(藍色包包)，背景(藍色大海)去達到統一色調的效果，再不行就是調色，將白色的雪調成偏藍色的。

有哪些特徵		
		
物件	背景	調色

根據上述觀察，我希望可以做出一款 APP ，讓使用者可以上傳照片並分類色調，當要發文時，可以從分類後頁面找到屬於此色調的照片。

設計說明

主題: 照片分類 APP

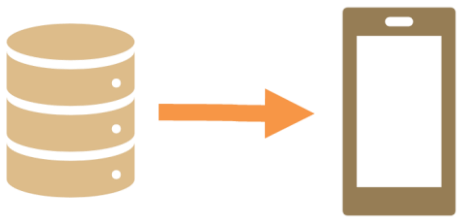
目標: 1.上傳照片 2.將照片依色系分類

用途: 1.方便使用者整理照片 2.社群網站版面/發文

● 首頁

功能: 顯示最近上傳的照片

作法: 從資料庫 GET 圖片名稱(以時間排序)，從本機取出對應照片，顯示在首頁。

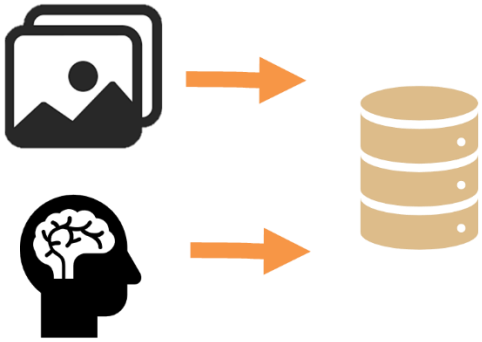


● 照片上傳頁面

功能:

- 照片上傳至後端
- 照片色系分類模型

作法:從相簿選擇圖片，顯示在畫面，POST 圖片資訊到資料庫，執行辨識程式，將結果傳入資料庫。

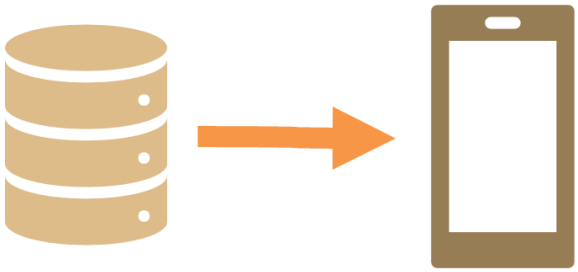


● 照片分類頁面

功能:

- 資料庫取出照片
- 依照照片數量動態顯示

作法:頁面分類成 gray、warm 、cool 三類 從資料庫根據頁面分類 GET 圖片資訊，根據查詢結果筆數，可以動態顯示在畫面上



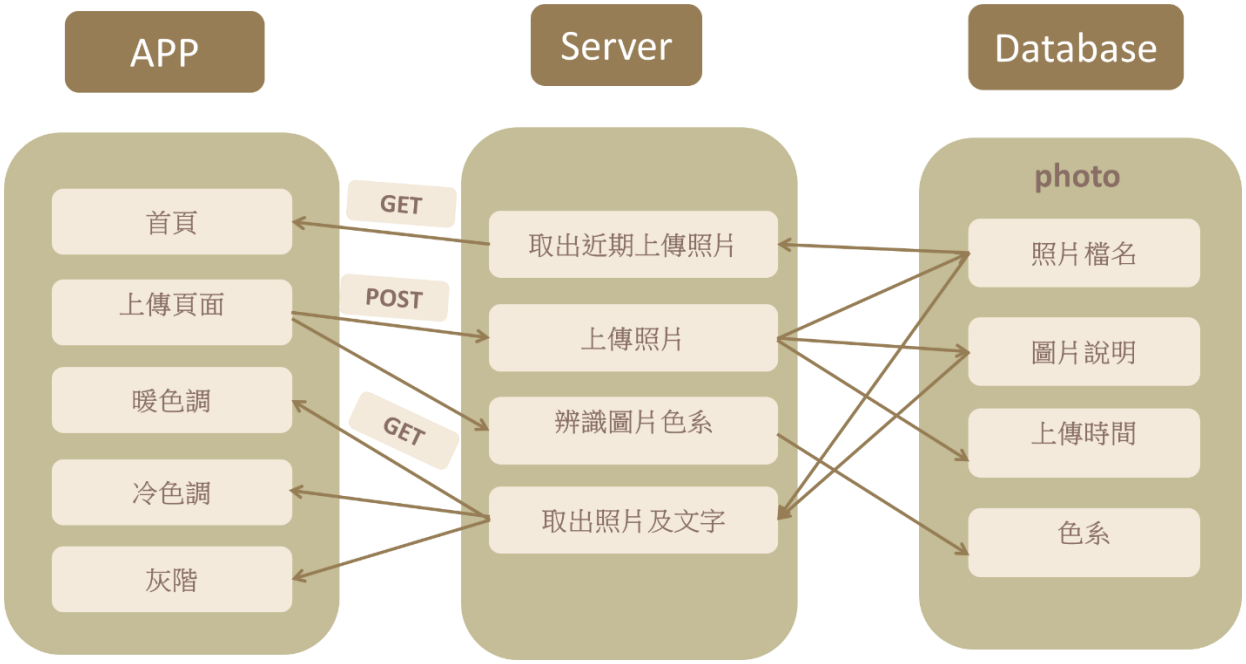
資料庫設計

資料表名稱: photo

欄位名稱	資料型態	是否為空值	意義
Imagetitle	varchar	否	圖片名稱
content	varchar	否	圖片說明
upload_date	datetime	否	上傳時間
color	varchar	是	分類結果 (gray、warm 、cool)

AI 分類色系後才會將結果存入資料庫

系統架構



開發流程

1. Navagation drawer

主程式： MainActivity.kt

Layout ： activity_main.xml

說明：

整個 APP 共有 5 個頁面，分別是

1. **首頁**
2. **上傳頁面**
3. **灰階**
4. **暖色調**
5. **冷色調**

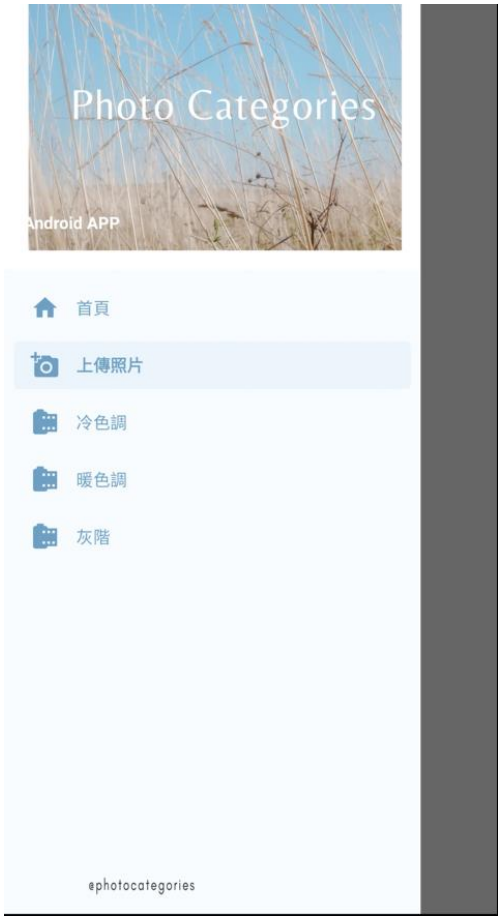
使用 canva 繪製 APP LOGO ，整體以藍色系為主要設計

程式碼：

```
class MainActivity : AppCompatActivity(), NavigationView.OnNavigationItemSelectedListener {
    private lateinit var drawerLayout: DrawerLayout

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)


        drawerLayout = findViewById<DrawerLayout>(R.id.drawer_layout)
        val toolbar = findViewById<Toolbar>(R.id.toolbar)
        setSupportActionBar(toolbar)
        val navigationView = findViewById<NavigationView>(R.id.nav_view)
        navigationView.setNavigationItemSelectedListener(this)
        val toggle = ActionBarDrawerToggle( activity: this, drawerLayout, toolbar, "Open Navigation Drawer", "Close Navigation Drawer")
        drawerLayout.addDrawerListener(toggle)
        toggle.syncState()
        if (savedInstanceState == null) {
            supportFragmentManager.beginTransaction()
                .replace(R.id.fragment_container, HomeFragment()).commit()
            navigationView.setCheckedItem(R.id.nav_home)
        }

    }
```

```
override fun onNavigationItemSelected(item: MenuItem): Boolean {
    when (item.itemId) {
        R.id.nav_home -> supportFragmentManager.beginTransaction()
            .replace(R.id.fragment_container, HomeFragment()).commit()
        R.id.nav_upload -> supportFragmentManager.beginTransaction()
            .replace(R.id.fragment_container, uploadFragment()).commit()
        R.id.nav_cold -> supportFragmentManager.beginTransaction()
            .replace(R.id.fragment_container, coolFragment()).commit()
        R.id.nav_warm -> supportFragmentManager.beginTransaction()
            .replace(R.id.fragment_container, warmFragment()).commit()
        R.id.nav_gray -> Toast.makeText( context: this,  text: "Logout!", Toast.LENGTH_SHORT).show()
    }
    drawerLayout.closeDrawer(GravityCompat.START)
    return true
}

override fun onBackPressed() {
    if (drawerLayout.isDrawerOpen(GravityCompat.START)) {
        drawerLayout.closeDrawer(GravityCompat.START)
    } else {
        onBackPressedDispatcher.onBackPressed()
    }
}
```

實作成果:



2.首頁

主程式: HomeFregment.kt

Layout ： fregment_home.xml

使用功能 ： Image Slider

說明:

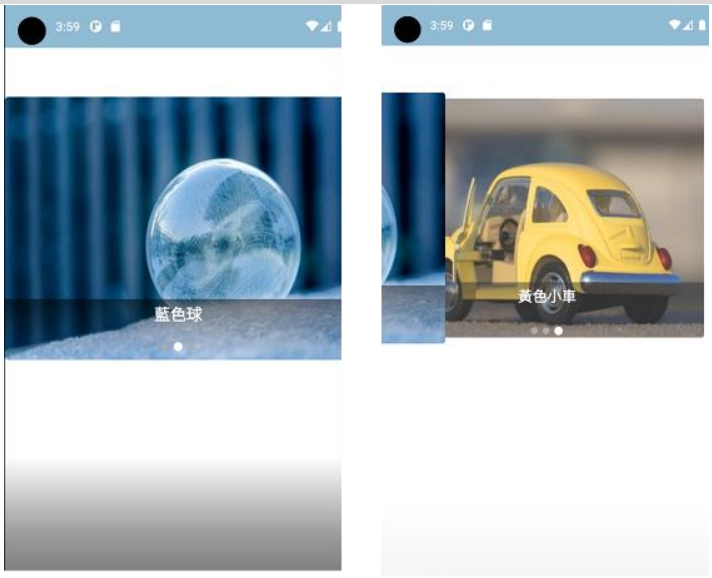
從資料庫 GET 圖片名稱(以時間排序)，從本機取出對應照片顯示在首頁的 Image Slider

程式碼:

name: 圖片名稱 text: 圖片說明

```
if (imageSlider != null) {
    val imageList = ArrayList<SlideModel>()
    //imageList.add(SlideModel("圖片網址", "Title名稱"))
    val resourceId = resources.getIdentifier(name[1], defType: "drawable", requireContext().packageName)
    val resourceId1 = resources.getIdentifier(name[2], defType: "drawable", requireContext().packageName)
    val resourceId2 = resources.getIdentifier(name[3], defType: "drawable", requireContext().packageName)

    imageList.add(
        SlideModel(
            resourceId,
            text[1]
        )
    )
    imageList.add(
        SlideModel(
            resourceId1 ,
            text[2]
        )
    )
    imageList.add(SlideModel(resourceId2 , text[3]))

    imageSlider?.setImageList(imageList, ScaleTypes.FIT)
    imageSlider?.setSlideAnimation(AnimationTypes.DEPTH_SLIDE)
    imageSlider?.startSliding( changeablePeriod: 3000)
```

實作成果:



照片以 Image Slider 方式顯示

3.照片上傳頁面

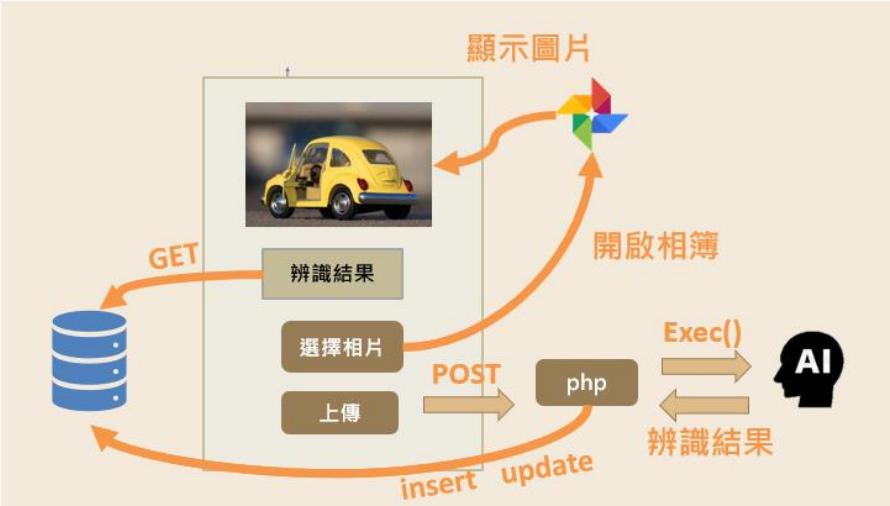
主程式: uploadFregment.kt

Layout ： fregment_upload.xml

說明:

從相簿選擇圖片，顯示在畫面，POST 圖片資訊到資料庫，執行辨識程式，將結果傳入資料庫

步驟示意圖:



程式碼:

```
imageView = view.findViewById(R.id.imageView)  // 取得ImageView元件
content = view.findViewById(R.id.editTextTextPersonName)  // 取得EditText元件
color = view.findViewById(R.id.textView2)  // 取得TextView元件
val selectImageButton = view.findViewById<Button>(R.id.selectImageButton)  // 取得選擇圖片按鈕元件
val uploadImageButton = view.findViewById<Button>(R.id.uploadImageButton)  // 取得上傳圖片按鈕元件

selectImageButton.setOnClickListener { it: View!
    openGallery()  // 點擊選擇圖片按鈕時執行openGallery函式
}

selectImageButton.setOnClickListener { it: View!
    val intent = Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI)
    startActivityForResult(
        intent,
        PICK_IMAGE_REQUEST
    )  // 點擊選擇圖片按鈕時執行選擇圖片的Intent，並啟動Activity以選擇圖片
}
```

```
    uploadImageButton.setOnClickListener { it: View!
        val imageBitmap = getBitmapFromImageView(imageView)  // 從ImageView獲取Bitmap圖像
        val contentText = content.text.toString()  // 獲取EditText中的文字內容
        println("postAccount-------$contentText")
        if (imageBitmap != null && contentText != "") {  // 確保圖片和文字內容都不為空
            uploadImage(imageBitmap, contentText)  // 呼叫上傳圖片的函式，傳入圖片和文字內容
        } else if (imageBitmap == null && contentText == "") {
            Log.e( tag: "UploadImage",  msg: "未選擇圖片及未填入文字說明")  // 如果圖片和文字內容都為空，輸出錯誤訊息
        } else if (imageBitmap == null) {
            Log.e( tag: "UploadImage",  msg: "未選擇圖片")  // 如果只有圖片為空，輸出錯誤訊息
        } else {
            Log.e( tag: "UploadImage",  msg: "未填入文字說明")  // 如果只有文字內容為空，輸出錯誤訊息
        }
    }

    return view  // 回傳視圖
}

private fun openGallery() {
    val intent = Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI)
    startActivityForResult(intent, PICK_IMAGE_REQUEST)  // 開啟圖片庫的Intent，並啟動Activity以選擇圖片
}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    if (requestCode == PICK_IMAGE_REQUEST && resultCode == Activity.RESULT_OK && data != null) {
        val selectedImageUri = data.data
        imageView.setImageURI(selectedImageUri)  // 設定選擇的圖片為ImageView的圖像
    }
}
```

```
private fun getBitmapFromImageView(imageView: ImageView): Bitmap? {
    imageView.isDrawingCacheEnabled = true
    imageView.buildDrawingCache()
    val bitmap = Bitmap.createBitmap(imageView.drawingCache)  // 從ImageView的繪圖緩存中獲取Bitmap圖像
    imageView.isDrawingCacheEnabled = false
    return bitmap
}

private fun uploadImage(bitmap: Bitmap, string: String) {
    val baos = ByteArrayOutputStream()
    bitmap.compress(Bitmap.CompressFormat.JPEG,  quality: 100, baos)  // 將Bitmap圖像壓縮為JPEG格式
    val imageBytes = baos.toByteArray()  // 將壓縮後的圖像轉換為位元組數組
    filename.inc()
    print(filename)

    val requestBody = MultipartBody.Builder()
        .setType(MultipartBody.FORM)
        .addFormDataPart(
            name: "image",
            filename: "a" + filename.toString() + ".jpg",
            imageBytes.toRequestBody("image/jpeg".toMediaTypeOrNull())  // 將圖像位元組數組轉換為RequestBody
        )
        .addFormDataPart( name: "content",  value: "${string}")  // 添加文字內容為RequestBody
        .build()

    val serverUrl = "http://10.123.8.252/saveimage/upload_identify.php"
    val request = Request.Builder()
        .url(serverUrl)
        .post(requestBody)
        .build()
```

```
val client = OkHttpClient.Builder()
    .connectTimeout(timeout, java.util.concurrent.TimeUnit.SECONDS)
    .readTimeout(timeout, java.util.concurrent.TimeUnit.SECONDS)
    .build()

GlobalScope.launch(Dispatchers.IO) { this: CoroutineScope
    try {
        val response = client.newCall(request).execute()  // 發送HTTP請求並獲取響應
        if (response.isSuccessful) {  // 如果響應成功
            requireActivity().runOnUiThread {
                Toast.makeText(requireContext(), text: "圖片上傳成功", Toast.LENGTH_SHORT)
                    .show()  // 在主線程中顯示上傳成功的訊息
            }

            val urlBuilder =
                "http://10.123.8.252/saveimage/read.php".toHttpUrlOrNull()?.newBuilder()
                    ?.addQueryParameter( name: "filename",  value: "a" + filename.toString() + ".jpg")
            val url = urlBuilder?.build().toString()
            val request = Request.Builder()
                .url(url)
                .build()
            client.newCall(request).enqueue(object : Callback {
                override fun onFailure(call: Call, e: IOException) {
                    e.printStackTrace()
                }

                override fun onResponse(call: Call, response: Response) {
                    println("123")
                    if (response.isSuccessful) {  // 如果響應成功
                        println("456")
                        val responseBody = response.body?.string()  // 獲取響應的內容

                        println("-----${responseBody}")

                        requireActivity().runOnUiThread {
                            // 在主線程中更新UI
                            color.append("\n$responseBody")  // 將響應的內容追加到TextView中
                        }
                    } else {
                        println("789")
                        println("Request failed")
                        requireActivity().runOnUiThread {
                            color.text = "資料錯誤" // 在主線程中更新UI，顯示資料錯誤的訊息
                        }
                    }
                    println("000")
                }
            })
        } else {
            requireActivity().runOnUiThread {
                Toast.makeText(requireContext(), text: "圖片上傳失敗", Toast.LENGTH_SHORT)
                    .show()  // 在主線程中顯示上傳失敗的訊息
            }
        }
    } catch (e: IOException) {
        e.printStackTrace()
        requireActivity().runOnUiThread {
            Toast.makeText(requireContext(), text: "網路錯誤", Toast.LENGTH_SHORT)
                .show()  // 在主線程中顯示網路錯誤的訊息
        }
    }
}
}
```

實作成果:

按下上傳後資料庫新增 a1001. jpg 此筆資料，再經果 AI 辨識將顏色結果 update 到資料庫

<table>
<tr><th colspan="2">APP 介面</th></tr>
<tr><td colspan="2">
<table>
<tr><th>imgtitle</th><th>upload_date</th><th>content</th><th>color
顏色</th></tr>
<tr><td>a1001.jpg</td><td>2023-06-12 12:00:54</td><td>yy</td><td>cool</td></tr>
<tr><td>a1003.jpg</td><td>2023-06-12 12:21:56</td><td>456</td><td>cool</td></tr>
<tr><td>a1006.jpg</td><td>2023-06-08 06:29:07</td><td>黃色小車</td><td>warm</td></tr>
<tr><td>a1009.jpg</td><td>2023-06-08 17:20:03</td><td>橙色機車</td><td>warm</td></tr>
</table>
</td></tr>
<tr><td colspan="2">資料庫內容</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>照片上傳至 server</td><td>辨識後從資料庫取得結果</td></tr>
</table>

4.照片分類頁面

主程式:

warmFregment.kt

coolFregment.kt

grayFregment.kt

Layout ：

fregment_warm.xml

fregment_cool.xml

fregment_gray.xml

說明:

頁面分類成 gray、warm 、cool 三類

從資料庫根據頁面分類 GET 圖片資訊，可以根據查詢結果筆數，動態顯示在畫面上

主要功能

程式碼:

```
private val content_items = mutableListOf<HashMap<String,String>>()

val warmLayoutId: Int = R.layout.fragment_warm
val timeout = 30L // 设置超时时间为30秒
override fun onCreateView(
    inflater: LayoutInflater,
    container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    val view = inflater.inflate(R.layout.fragment_warm, container,  attachToRoot: false)

    val client = OkHttpClient.Builder()
    .connectTimeout(timeout, java.util.concurrent.TimeUnit.SECONDS)
    .readTimeout(timeout, java.util.concurrent.TimeUnit.SECONDS)
    .build()
    //GET color=warm 的所有圖片資料
    val urlBuilder = "http://10.123.8.252/saveimage/read_photo.php".toHttpUrlOrNull()?.newBuilder()?.addQueryParameter( name: "color",  value: "warm")
    val url = urlBuilder?.build().toString()
    val request = Request.Builder()
        .url(url)
        .build()
    client.newCall(request).enqueue(object : Callback {
        override fun onFailure(call: Call, e: IOException) {
            e.printStackTrace()
        }
```

```
    }
//GET 後的結果
override fun onResponse(call: Call, response: Response) {
    println("123")
    if (response.isSuccessful) {
        println("456")
        val responseBody = response.body?.string()

        println("-----${responseBody}")

        val jsonObject = JSONObject(responseBody)
        val dataArray = jsonObject.getJSONArray( name: "data")


        //將圖片名稱及說明存入map資料結構中
        for (i in 0 ≤ until < dataArray.length()) {
            println("-----111")
            val map = HashMap<String, String>()
            val chatObject = dataArray.getJSONObject(i)
            val imgtitle = chatObject.getString( name: "imgtitle")
            val content = chatObject.getString( name: "content")
            println("-----${imgtitle}")
            println("-----${content}")
            map["img"] = imgtitle
            map["text"] = content
            content_items.add(map)

        }
    } else {
        println("789")
        println("Request failed")
          requireActivity().runOnUiThread {
              color.text = "資料錯誤" // Update UI on the main thread
          }
    }
    println("000")

    println("--2--")

    val parentLayout: LinearLayout = view.findViewById(R.id.parentLayout)
    println("--3--")
    //將每筆資料layout到版面上
    for (item in content_items) {
        val linearLayout = LinearLayout(requireContext())
        linearLayout.orientation = LinearLayout.HORIZONTAL
        linearLayout.layoutParams = LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.MATCH_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT
        )
        println("--4--")

        val name = item["img"]
        val extension = ".jpg" // Change the extension to match your image file's extension

        // 获取图片资源ID
        val resourceId = resources.getIdentifier(name, defType: "drawable", requireContext().packageName)
        Log.d( tag: "Resource ID", msg: "The resource ID for $name is $resourceId")
```

```
// 创建 ImageView 并设置图片资源
val imageView = ImageView(requireContext())
imageView.setImageResource(resourceId)
// 设置 ImageView 的宽高
val imageLayoutParams = LinearLayout.LayoutParams(MATCH_PARENT,  height: 500)
imageView.layoutParams = imageLayoutParams

// 创建 TextView 并设置文本
val textView = TextView(requireContext())
textView.text = item["text"]
// 设置 TextView 的宽高
val textLayoutParams = LinearLayout.LayoutParams(
    LinearLayout.LayoutParams.WRAP_CONTENT,
    LinearLayout.LayoutParams.WRAP_CONTENT
)
textLayoutParams.marginStart = 16 // 设置 TextView 和 ImageView 之间的间距
textView.layoutParams = textLayoutParams
println("--3--")
// 创建包含 ImageView 和 TextView 的 LinearLayout
val containerLayout = LinearLayout(requireContext())
containerLayout.orientation = LinearLayout.HORIZONTAL
containerLayout.layoutParams = LinearLayout.LayoutParams(
    LinearLayout.LayoutParams.MATCH_PARENT,
    LinearLayout.LayoutParams.WRAP_CONTENT
)
containerLayout.addView(imageView)
containerLayout.addView(textView)
println("--2--")
```

```
// 创建包含 ImageView 和 TextView 的 LinearLayout
val containerLayout = LinearLayout(requireContext())
containerLayout.orientation = LinearLayout.HORIZONTAL
containerLayout.layoutParams = LinearLayout.LayoutParams(
    LinearLayout.LayoutParams.MATCH_PARENT,
    LinearLayout.LayoutPara
)
containerLayout.addView(imageView)
containerLayout.addView(textView)
println("--2--")
// 将包含 ImageView 和 TextView 的 LinearLayout 添加到父布局
parentLayout.addView(containerLayout)
println("--1--")
```

Fetching Documentation...

實作成果:

Warm 查詢後只有兩筆資料，所以只顯示兩張照片

總結心得

一開始在猶豫是否要用已經寫過的購物網站，不過因為想要嘗試將 APP 和人工智慧結合，沒有想到購物網站可以有什麼可以結合的部分，於是決定嘗試新題目，從一開始想分類底片，因為不同底片有不同色調，後來發現實作上有點困難，於是想到 IG 排版色調統一，從這個方向著手，因為是從 0 開始，也沒有人做過類似的題目，所以剛開始卡關很久，不果相較於 APP 部分，辨識這邊的困難比較還好，雖然一開始在安裝環境時遇到好多問題，而且還繞了一圈 CPU 改 GPU 後來所小圖片像素在 CPU 上跑，但是確定好 k-means 模型後就蠻順利的，這次是我第一次自己一個人開發一份專案，很有成就感，因為以前都是小組做專題，所以 debug 或是構想都可以一起討論，但這次所有事情都要自己來，變成依賴網路資料、Chat GPT，再將得到的資訊過濾，因為我發現 Chat GPT 回答不一定都是正確的。

這次專題我學到如何尋找題目，並將構想修飾成可以實作的題目，學會如何下查詢指令，都對以後研究所如何找論文題目很有幫助，老師在過程中也抽空幫我不少忙，給我一些提點，很謝謝老師，也祝老師在彰師教書可以順順利利。