```kotlin
package com.example.timewellspent

import android.content.Intent
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.PopupMenu
import android.widget.TextView
import androidx.constraintlayout.widget.ConstraintLayout
import androidx.recyclerview.widget.RecyclerView
import com.backendless.Backendless
import com.backendless.async.callback.AsyncCallback
import com.backendless.exceptions.BackendlessFault
import com.google.android.material.floatingactionbutton.FloatingActionButton
import java.text.DateFormat
import java.text.SimpleDateFormat


class GameAdapter(var gameList: MutableList<GameEntry>) : RecyclerView.Adapter<GameAdapter.ViewHolder>() {

    class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val textViewName: TextView
        val textViewDate: TextView
        val textViewMoneySpent: TextView
        val textViewTimeSpent: TextView
        val textViewEmotion: TextView
        val layout: ConstraintLayout

        init {
            textViewName = itemView.findViewById(R.id.textView_gameEntry_name)
            textViewDate = itemView.findViewById(R.id.textView_gameEntry_date)
            textViewMoneySpent = itemView.findViewById(R.id.textView_gameEntry_moneySpent)
            textViewTimeSpent = itemView.findViewById(R.id.textView_gameEntry_timeSpent)
            textViewEmotion = itemView.findViewById(R.id.textView_gameEntry_emotion)
            layout = itemView.findViewById(R.id.layout_gameEntry)
        }

    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        val view =
            LayoutInflater.from(parent.context).inflate(R.layout.item_game_entry, parent, false)
        val holder = ViewHolder(view)
        return holder
    }

    override fun getItemCount(): Int {
        return gameList.size
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val game = gameList[position]
        val context = holder.layout.context
        holder.textViewName.text = game.name
        // TODO: format the date nicely to show just the day month and year
        val format: DateFormat = SimpleDateFormat("EEEE, MMMM d, yyyy")
        val formatted: String = format.format(game.datePlayed)
        holder.textViewDate.text = formatted
        // TODO: format the time to show it in hours and minutes
        holder.textViewTimeSpent.text = "${game.elapsedTime/60} hrs ${game.elapsedTime%60} mins"
        // TODO: format the money nicely to show it like $5.99
        holder.textViewMoneySpent.text = "$${game.moneySpent/100}.${game.moneySpent%100}"
        if(game.moneySpent%100.toString().length < 2)
            holder.textViewMoneySpent.text = "$${game.moneySpent/100}.${game.moneySpent%100}0"
        // TODO: verify this works in displaying the emoji
        holder.textViewEmotion.text = try {
            GameEntry.EMOTION.valueOf(game.emotion).emoji
        } catch (ex: IllegalArgumentException) {
            "¯\\_(ツ)_/¯"
        }

        holder.layout.isLongClickable = true
        holder.layout.setOnLongClickListener {
            // the holder.textViewBorrower is the textView that the PopMenu will be anchored to
            val popMenu = PopupMenu(context, holder.textViewName)
            popMenu.inflate(R.menu.menu_game_list_context)
            popMenu.setOnMenuItemClickListener {
                when(it.itemId) {
                    R.id.menu_game_delete -> {
                        deleteFromBackendless(position)
                        true
                    }
                    else -> true
                }
            }
            popMenu.show()
            true
        }

        holder.layout.setOnClickListener {
            val context = holder.layout.context
            val detailIntent = Intent(context, GameDetailActivity::class.java)
            detailIntent.putExtra(GameDetailActivity.EXTRA_GAME_ENTRY, game)
            context.startActivity(detailIntent)
        }


    }
    private fun deleteFromBackendless(position: Int) {
        Log.d("GameAdapter", "deleteFromBackendless: Trying to delete ${gameList[position]}")
        // put in the code to delete the item using the callback from Backendless
        Backendless.Data.of(GameEntry::class.java).remove(gameList[position],
            object : AsyncCallback<Long?> {
                override fun handleResponse(response: Long?) {
                    // Contact has been deleted. The response is the
                    // time in milliseconds when the object was deleted
                    Log.d("Game Adapter", "handleResponse:${response}")
                    gameList.remove(gameList[position])
                    notifyDataSetChanged()

                }

                override fun handleFault(fault: BackendlessFault) {
                    // an error has occurred, the error code can be
                    // retrieved with fault.getCode()
                    Log.d("Game Adapter", "handleFault:${fault.message}")
                }
            })
        // in the handleResponse, we'll need to also delete the item from the sleepList
        // and make sure that the recyclerview is updated
    }
}
```

```kotlin
package com.example.timewellspent

import android.content.Intent
import android.os.Bundle
import android.util.Log
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import androidx.recyclerview.widget.LinearLayoutManager
import com.backendless.Backendless
import com.backendless.BackendlessUser
import com.backendless.async.callback.AsyncCallback
import com.backendless.exceptions.BackendlessFault
import com.backendless.persistence.DataQueryBuilder
import com.example.timewellspent.databinding.ActivityGameListBinding


class GameListActivity : AppCompatActivity() {

    private lateinit var binding: ActivityGameListBinding
    private lateinit var adapter: GameAdapter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        binding = ActivityGameListBinding.inflate(layoutInflater)
        setContentView(binding.root)
        ViewCompat.setOnApplyWindowInsetsListener(binding.root) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }

        // make backendless call to retrieve all data

        // want the userid of the logged in user to match the ownerId of the object
//        val userId = Backendless.UserService.CurrentUser().userId
//        val userId = intent.getStringExtra(LoginActivity.EXTRA_USER_ID)
//
//        // ownerId = 'userId'
//        val whereClause = "ownerId = '$userId'"
//        val queryBuilder = DataQueryBuilder.create()
//        queryBuilder.setWhereClause(whereClause)

//        Backendless.Data.of(GameEntry::class.java).find(queryBuilder, object : AsyncCallback<MutableList<
GameEntry>> {
//            override fun handleResponse(foundGameEntries: MutableList<GameEntry>) {
//                // all GameEntry instances have been found
//                Log.d("GameListActivity","handleResponse: $foundGameEntries")
//                adapter = GameAdapter(foundGameEntries)
//                binding.recyclerViewGameListActivityList.adapter = adapter
//                binding.recyclerViewGameListActivityList.layoutManager = LinearLayoutManager(this@
GameListActivity)
//            }
//
//            override fun handleFault(fault: BackendlessFault) {
//                // an error has occurred, the error code can be retrieved with fault.getCode()
//                Log.d("GameListActivity","handleFault: ${fault.message}")
//            }
//        })

//        binding.fabGameListNewEntry.setOnClickListener {
//            val context = binding.fabGameListNewEntry.context
//            val detailIntent = Intent(context, GameDetailActivity::class.java)
//            context.startActivity(detailIntent)
//        }
```

```kotlin
        // in the handleResponse, get the list of data and constructor the adapter & apply to the reyclerview

    }

    override fun onStart() {
        super.onStart()
        val userId = intent.getStringExtra(LoginActivity.EXTRA_USER_ID)

        // ownerId = 'userId'
        val whereClause = "ownerId = '$userId'"
        val queryBuilder = DataQueryBuilder.create()
        queryBuilder.setWhereClause(whereClause)

        Backendless.Data.of(GameEntry::class.java).find(queryBuilder, object : AsyncCallback<MutableList<GameEntry>> {
            override fun handleResponse(foundGameEntries: MutableList<GameEntry>) {
                // all GameEntry instances have been found
                Log.d("GameListActivity","handleResponse: $foundGameEntries")
                adapter = GameAdapter(foundGameEntries)
                binding.recyclerViewGameListActivityList.adapter = adapter
                binding.recyclerViewGameListActivityList.layoutManager = LinearLayoutManager(this@GameListActivity)
            }

            override fun handleFault(fault: BackendlessFault) {
                // an error has occurred, the error code can be retrieved with fault.getCode()
                Log.d("GameListActivity","handleFault: ${fault.message}")
            }
        })

        binding.fabGameListNewEntry.setOnClickListener {
            val context = binding.fabGameListNewEntry.context
            val detailIntent = Intent(context, GameDetailActivity::class.java)
            context.startActivity(detailIntent)
        }
    }

}
```

```kotlin
package com.example.timewellspent

import android.R
import android.os.Bundle
import android.util.Log
import android.widget.ArrayAdapter
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
import com.backendless.Backendless
import com.backendless.async.callback.AsyncCallback
import com.backendless.exceptions.BackendlessFault
import com.example.timewellspent.databinding.ActivityGameDetailBinding
import com.google.android.material.datepicker.MaterialDatePicker
import kotlinx.coroutines.NonCancellable.start
import java.text.DateFormat
import java.text.SimpleDateFormat
import java.util.Date
import java.util.Locale


class GameDetailActivity : AppCompatActivity() {

    companion object {
        val TAG = "GameDetailActivity"
        val EXTRA_GAME_ENTRY = "game entry"
    }

    private lateinit var binding: ActivityGameDetailBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        binding = ActivityGameDetailBinding.inflate(layoutInflater)
        setContentView(binding.root)
        ViewCompat.setOnApplyWindowInsetsListener(binding.root) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }

        val gameEntry = intent.getParcelableExtra<GameEntry>(EXTRA_GAME_ENTRY) ?: GameEntry()

        binding.editTextGameDetailName.setText(gameEntry.name)
        binding.editTextGameDetailMoneySpent.setText("${gameEntry.moneySpent/100}.${gameEntry.moneySpent%
100}")
        binding.sliderGameDetailTimeSpent.value = gameEntry.elapsedTime.toFloat()
        val format: DateFormat = SimpleDateFormat("yyyy-MM-dd", Locale.US)
        binding.textViewGameDetailDate.text = format.format(gameEntry.datePlayed)

        var spinnerItems = GameEntry.EMOTION.entries.map { Pair(it.emoji, it.name) }
        binding.spinnerGameDetailEmotion.adapter =
            ArrayAdapter(this, R.layout.simple_spinner_dropdown_item, spinnerItems.map { it.first })
        Log.d(TAG, "onCreate: $gameEntry  ${gameEntry.emotion} ${ spinnerItems.map { it.second }.indexOf(
gameEntry.emotion)}")
        var position = spinnerItems.map { it.second }.indexOf(gameEntry.emotion)
        if(position < 0) {
            position = 0
        }
        binding.spinnerGameDetailEmotion.setSelection(position)

        binding.textViewGameDetailDate.setOnClickListener {
            val selection = binding.textViewGameDetailDate.text.toString()
            val date: Date = format.parse(selection)
            val datePicker = MaterialDatePicker.Builder.datePicker()
                .setSelection(date.time) // requires milliseconds
                .setTitleText("Select a Date")
                .build()

            datePicker.addOnPositiveButtonClickListener { millis ->
                val newDate = Date(millis+24*60*60*1000)
                binding.textViewGameDetailDate.setText(format.format(newDate))
            }

            datePicker.show(supportFragmentManager,"date picker")

        }

        binding.buttonGameDetailSave.setOnClickListener {
            gameEntry.ownerId = Backendless.UserService.CurrentUser().userId
            gameEntry.name = binding.editTextGameDetailName.text.toString()
            // for money spent, remove the leading $
            val moneyString = binding.editTextGameDetailMoneySpent.text.toString()
            var cents = 0
            if(!moneyString.contains(".")) {
                cents = moneyString.toInt() * 100
            } else {
                var dollarsAndCents = moneyString.split(".")
                var dollarString = dollarsAndCents[0]
                var centsString = dollarsAndCents[1]
                if(dollarString.isNotEmpty()) {
                    cents += dollarString.toInt() * 100
                }
                if(centsString.isNotEmpty()) {
                    if(centsString.length > 2) {
                        centsString = centsString.substring(0,2)
                    } else if(centsString.length == 1) {
                        centsString += "0"
                    }
                    cents += centsString.toInt()
                }
            }
            gameEntry.moneySpent = cents
            gameEntry.elapsedTime = binding.sliderGameDetailTimeSpent.value.toInt()
            gameEntry.datePlayed =  format.parse(binding.textViewGameDetailDate.text.toString()) ?: Date()
            gameEntry.emotion =  GameEntry.EMOTION.entries.find { it.emoji == binding.
spinnerGameDetailEmotion.selectedItem.toString()}!!.name
            saveToBackendless(gameEntry)
            finish()
        }
    }



    private fun saveToBackendless(gameEntry: GameEntry) {
        // code here to save to backendless
        Backendless.Data.of(GameEntry::class.java).save(gameEntry, object : AsyncCallback<GameEntry?> {
            override fun handleResponse(response: GameEntry?) {
                // new Contact instance has been saved
                Log.d(TAG, "handleResponse: Saved to Backendless")
            }

            override fun handleFault(fault: BackendlessFault) {
                // an error has occurred, the error code can be retrieved with fault.getCode()
                Log.d(TAG, "handleFault: ${fault.message}")
            }
        })
    }




}
```