package com.sharehub.sharehub.sharehub

import android.annotation.SuppressLint

import android.app.AlertDialog

import android.content.Intent

import android.os.Bundle

import android.text.InputType

import android.view.View

import android.widget.\*

import com.loopj.android.http.AsyncHttpClient

import com.loopj.android.http.JsonHttpResponseHandler

import com.loopj.android.http.RequestParams

import cz.msebera.android.httpclient.Header

import org.json.JSONObject

import java.util.\*

class Profile : ShareHubActivityLayoutBase() {

@SuppressLint("SetTextI18n")

override fun onCreate(savedInstanceState: Bundle?) {

setContentView(R.layout.profile)

super.onCreate(savedInstanceState)

val bundle = intent.extras

if (bundle == null) {

replaceActivity(Intent(this, ProfileLauncher::class.java))

return

}

val profileData = bundle.getSerializable("profileData") as ProfileLauncher.ProfileData

val profileUsernameTitle: TextView = findViewById(R.id.profile\_username\_title)

val profileNfcDisplay: TextView = findViewById(R.id.profile\_nfc\_display)

val profileNfcButton: Button = findViewById(R.id.profile\_nfc\_button)

val profileBoxes: LinearLayout = findViewById(R.id.profile\_boxes)

val profileCommunities: LinearLayout = findViewById(R.id.profile\_communities)

val profileCommunitiesButton: Button = findViewById(R.id.profile\_communities\_button)

var profileCommunitiesShows = false

var boxToGive: Int? = null

if (profileData.nfcId == null) {

profileNfcButton.text = "initialize a new nfc"

profileNfcDisplay.text = "no nfc initialized"

profileNfcButton.setOnClickListener { view ->

run {

Statics.requestNfc(this, { id ->

run {

val builder = Statics.waiterDialog(this)

builder.show()

val client = AsyncHttpClient()

val params = RequestParams()

params.add("key", Statics.UserKey)

params.add("nfcId", id)

fun reload(key: String?) {

ProfileLauncher.profileDataMap[Statics.Username]!!.nfcId = key

val intent = Intent(this@Profile, ProfileLauncher::class.java)

intent.putExtra("user", Statics.Username)

replaceActivity(intent)

}

client.post(Constants.UserGateway.RegisterNfc, params, object : JsonHttpResponseHandler() {

override fun onSuccess(statusCode: Int, headers: Array<out Header>?, response: JSONObject) {

if (!response.getBoolean("success")) {

Toast.makeText(this@Profile, "an error occurred", Toast.LENGTH\_LONG).show()

}

reload(id)

}

override fun onFailure(statusCode: Int, headers: Array<out Header>?, responseString: String?, throwable: Throwable?) {

Toast.makeText(this@Profile, "an http error occurred: " + throwable?.message, Toast.LENGTH\_LONG).show()

reload(null)

}

})

}

})

}

}

}

else {

profileNfcButton.text = "delete"

profileNfcDisplay.text = profileData.nfcId

profileNfcButton.setOnClickListener { view ->

run {

val client = AsyncHttpClient()

val params = RequestParams()

params.add("key", Statics.UserKey)

client.post(Constants.UserGateway.DeleteNfc, params, object : JsonHttpResponseHandler() {

override fun onSuccess(statusCode: Int, headers: Array<out Header>?, response: JSONObject) {

if (response.getBoolean("success")) {

ProfileLauncher.profileDataMap[Statics.Username]!!.nfcId = null

val intent = Intent(this@Profile, ProfileLauncher::class.java)

intent.putExtra("user", Statics.Username)

replaceActivity(intent)

}

else {

Toast.makeText(this@Profile, "an error occurred", Toast.LENGTH\_LONG).show()

}

}

override fun onFailure(statusCode: Int, headers: Array<out Header>?, responseString: String?, throwable: Throwable?) {

Toast.makeText(this@Profile, "http error: " + throwable?.message, Toast.LENGTH\_LONG).show()

}

})

}

}

}

run {

if (profileData.boxes!!.isEmpty()) {

val view = TextView(this)

view.text = "you have no boxes"

profileBoxes.addView(view)

}

else {

for (i in 0 until profileData.boxes!!.count()) {

val box: String = profileData.boxes!![i]

val boxKey: String = profileData.boxesKey!![i]

val entryView = LinearLayout(this)

val params = LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP\_CONTENT, LinearLayout.LayoutParams.WRAP\_CONTENT)

params.resolveLayoutDirection(LinearLayout.HORIZONTAL)

entryView.layoutParams = params

val mainTextView = TextView(this)

mainTextView.text = box

entryView.addView(mainTextView)

val renameButton = Button(this)

renameButton.text = "rename"

renameButton.setOnClickListener { view ->

run {

val builder = AlertDialog.Builder(this)

builder.setTitle("new name")

val input = EditText(this)

input.inputType = InputType.TYPE\_CLASS\_TEXT or InputType.TYPE\_TEXT\_VARIATION\_PASSWORD

builder.setView(input)

builder.setPositiveButton("OK") { dialog, which ->

run {

val context = this@run;

val renameClient = AsyncHttpClient()

val renameParams = RequestParams()

renameParams.add("userKey", Statics.UserKey)

renameParams.add("boxKey", boxKey)

renameParams.add("newName", input.text.toString())

renameClient.post(Constants.BoxGateway.Rename, renameParams, object : JsonHttpResponseHandler() {

override fun onSuccess(statusCode: Int, headers: Array<out Header>?, response: JSONObject) {

try {

if (response["success"] as Boolean) {

mainTextView.text = input.text.toString()

}

else {

Toast.makeText(context, "you have been logged out", Toast.LENGTH\_LONG).show()

Statics.logout(context)

context.replaceActivity(Intent(context, Login::class.java))

}

}

catch (e: Exception) {

Toast.makeText(context, "an exception occurred: " + e.message, Toast.LENGTH\_LONG).show()

}

}

override fun onFailure(statusCode: Int, headers: Array<out Header>?, responseString: String?, throwable: Throwable?) {

Toast.makeText(context, "2\_http request error: " + throwable?.message, Toast.LENGTH\_LONG).show()

}

})

}

}

builder.setNegativeButton("Cancel") { dialog, which -> dialog.cancel() }

builder.show()

}

}

entryView.addView(renameButton)

val giveButton = Button(this)

giveButton.text = "give"

giveButton.setOnClickListener { view ->

run {

if (boxToGive == null) {

boxToGive = i

if (profileCommunities.visibility == View.GONE) {

profileCommunitiesButton.callOnClick()

}

giveButton.text = "stop giving"

for (j in 0 until profileBoxes.childCount) {

if (i != j) {

profileBoxes.getChildAt(j).visibility = View.GONE

}

}

for (j in 0 until profileCommunities.childCount) {

(profileCommunities.getChildAt(j) as LinearLayout).getChildAt(2).visibility = View.VISIBLE

}

}

else {

boxToGive = null

giveButton.text = "give"

for (j in 0 until profileBoxes.childCount) {

profileBoxes.getChildAt(j).visibility = View.VISIBLE

}

for (j in 0 until profileCommunities.childCount) {

(profileCommunities.getChildAt(j) as LinearLayout).getChildAt(2).visibility = View.GONE

}

}

}

}

entryView.addView(giveButton)

profileBoxes.addView(entryView)

}

}

}

try {

if (profileData.communitiesName!!.isEmpty()) {

profileCommunitiesButton.visibility = View.GONE

val textView = TextView(this)

textView.text = "you are currently not in any communities"

profileCommunities.addView(textView)

}

else {

for (i in 0 until profileData.communitiesName!!.size) {

val entryView = LinearLayout(this)

val params = LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP\_CONTENT, LinearLayout.LayoutParams.WRAP\_CONTENT)

params.resolveLayoutDirection(LinearLayout.HORIZONTAL)

entryView.layoutParams = params

val textView = TextView(this)

textView.text = profileData.communitiesName!![i]

entryView.addView(textView)

val buttonView = Button(this)

buttonView.text = "view"

buttonView.setOnClickListener { view ->

run {

val communityIntent = Intent(this, CommunityLauncher::class.java)

communityIntent.putExtra("community", profileData.communitiesUniqueName!![i])

replaceActivity(communityIntent)

}

}

entryView.addView(buttonView)

val giveButtonView = Button(this)

giveButtonView.text = "give"

giveButtonView.setOnClickListener { view ->

run {

val waiter = Statics.waiterDialog(this)

waiter.show()

val client = AsyncHttpClient()

val httpParams = RequestParams()

httpParams.add("userKey", Statics.UserKey)

httpParams.add("boxKey", profileData.boxesKey!![boxToGive!!])

httpParams.add("to", profileData.communitiesUniqueName!![i])

httpParams.add("toType", "community")

fun reload() {

val boxesKey = profileData.boxesKey!!.toMutableList()

boxesKey.removeAt(boxToGive!!)

profileData.boxesKey = boxesKey.toTypedArray()

val boxes = profileData.boxes!!.toMutableList()

boxes.removeAt(boxToGive!!)

profileData.boxes = boxes.toTypedArray()

ProfileLauncher.profileDataMap[Statics.Username]!!.boxesKey = profileData.boxesKey

ProfileLauncher.profileDataMap[Statics.Username]!!.boxes = profileData.boxes

val intent = Intent(this@Profile, ProfileLauncher::class.java)

intent.putExtra("user", Statics.Username)

replaceActivity(intent)

}

client.post(Constants.BoxGateway.MoveBox, httpParams, object : JsonHttpResponseHandler() {

override fun onSuccess(statusCode: Int, headers: Array<out Header>?, response: JSONObject) {

if (!response.getBoolean("success")) {

Toast.makeText(this@Profile, "an error occurred: " + response.getString("message"), Toast.LENGTH\_LONG).show()

}

reload()

}

override fun onFailure(statusCode: Int, headers: Array<out Header>?, responseString: String?, throwable: Throwable?) {

Toast.makeText(this@Profile, "an http error occurred: " + throwable?.message, Toast.LENGTH\_LONG).show()

reload()

}

})

}

}

giveButtonView.visibility = View.GONE

entryView.addView(giveButtonView)

profileCommunities.addView(entryView)

}

}

profileCommunities.visibility = View.GONE

}

catch (e: Exception) {

}

profileCommunitiesButton.setOnClickListener { view ->

run {

profileCommunities.visibility = if (profileCommunitiesShows) {

profileCommunitiesButton.text = "show communities"

View.GONE

}

else {

profileCommunitiesButton.text = "hide communities"

View.VISIBLE

}

profileCommunitiesShows = !profileCommunitiesShows

}

}

profileUsernameTitle.text = profileData.username

}

}