class LoginFragment : Fragment() {

private var \_binding: FragmentLoginBinding? = null

private val binding: FragmentLoginBinding get() = \_binding!!

private var mCallbacksClient: PhoneAuthProvider.OnVerificationStateChangedCallbacks? = null

private var mResendingTokenClient: PhoneAuthProvider.ForceResendingToken? = null

private lateinit var mGoogleSignInClient: GoogleSignInClient

override fun onCreateView(

inflater: LayoutInflater, container: ViewGroup?,

savedInstanceState: Bundle?

): View {

\_binding = FragmentLoginBinding.inflate(inflater, container, false)

return binding.*root*

}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

super.onViewCreated(view, savedInstanceState)

initVars()

initAgreeText()

initGoogleSignVars()

binding.loginButton.setOnClickListener **{**

val phoneNumber = binding.phoneNumberEditText.*text*.toString()

if (phoneNumber.length == 9) {

binding.progressBar.*show*()

val options = PhoneAuthOptions.newBuilder(FirebaseAuth.getInstance())

.setPhoneNumber("+996$phoneNumber")

.setTimeout(60L, TimeUnit.*SECONDS*)

.setActivity(requireActivity())

.setCallbacks(mCallbacksClient!!)

.build()

PhoneAuthProvider.verifyPhoneNumber(options)

} else {

binding.phoneNumberEditText.setBackgroundResource(R.drawable.*back\_edit\_error*)

binding.textViewPhoneHint.setTextColor(

ContextCompat.getColor(requireContext(), R.color.*colorError*)

)

}

**}**

binding.googleButton.setOnClickListener **{**

signInWithGoogleLogin()

**}**

binding.toMainButton.setOnClickListener **{**

goToMain()

**}**

}

private fun initAgreeText() {

FirebaseFirestore.getInstance().collection("admin")

.document("agree").get().addOnSuccessListener **{**

val agreeText = **it**.getString("agreeText")

binding.buttonAgreement.setOnClickListener **{**

val dialog = CustomAgreements(agreeText ?: "text empty")

dialog.show(requireActivity().*supportFragmentManager*, "privacy policy")

**}**

**}**

}

private fun signInWithGoogleLogin() {

val intent = mGoogleSignInClient.*signInIntent*

startActivityForResult(intent, *RC\_GOOGLE\_SIGN\_IN*)

}

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

super.onActivityResult(requestCode, resultCode, data)

if (requestCode == *RC\_GOOGLE\_SIGN\_IN*) {

val task = GoogleSignIn.getSignedInAccountFromIntent(data)

try {

val account = task.getResult(ApiException::class.*java*)

binding.progressBar.*show*()

firebaseAuthWithGoogle(account)

} catch (e: ApiException) {

requireContext().*toast*(getString(R.string.*tryAgainError*) + " ${e.message}")

binding.progressBar.*hide*()

}

}

}

private fun firebaseAuthWithGoogle(account: GoogleSignInAccount?) {

val credential = GoogleAuthProvider.getCredential(account?.*idToken*, null)

val auth = FirebaseAuth.getInstance()

auth.signInWithCredential(credential).addOnCompleteListener **{**

if (**it**.*isSuccessful*) {

val user = FirebaseAuth.getInstance().*currentUser*

if (user != null)

saveUserData(user, account)

} else {

requireView().*showRegSnackBar*(getString(R.string.*tryAgainError*))

binding.progressBar.*hide*()

}

**}**

}

private fun saveUserData(user: FirebaseUser, account: GoogleSignInAccount?) {

FirebaseFirestore.getInstance().collection("users").document(user.*uid*).get()

.addOnSuccessListener **{**

val isProfileDone = **it**.getBoolean(*profileDone*)

if (isProfileDone != null && isProfileDone) {

goToMain()

} else {

val map = *mutableMapOf*<String, Any>()

map["userName"] = account?.*displayName*.*toString*()

map["userPhoto"] = account?.*photoUrl*.*toString*()

map["login"] = user.*email*.*toString*()

map["profileDone"] = true

map["userAddress"] = 0

map["userAddressText"] = ""

map["friends"] = 0

map["images"] = 0

map["uid"] = user.*uid*

map["bio"] = "-"

FirebaseFirestore.getInstance().collection("users").document(user.*uid*)

.set(map, SetOptions.merge()).addOnSuccessListener **{**

goToMain()

**}**

}

**}**

}

private fun goToMain() {

Intent(requireContext(), MainActivity::class.*java*).*let* **{** intent **->**

intent.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*)

requireActivity().finish()

startActivity(intent)

**}**

}

private fun signInWithPhone(authCredential: PhoneAuthCredential) {

binding.progressBar.*show*()

val firebaseAuth = FirebaseAuth.getInstance()

firebaseAuth.signInWithCredential(authCredential).addOnCompleteListener **{**

if (**it**.*isSuccessful*) {

val user = FirebaseAuth.getInstance().*currentUser*

if (user != null)

checkIfProfileDone(user, user.*phoneNumber* ?: "noPhone")

} else {

Log.d("LoginFragment", "signInWithPhone: ${**it**.*exception*}")

}

**}**

}

private fun checkIfProfileDone(user: FirebaseUser, login: String) {

FirebaseFirestore.getInstance().collection("users").document(user.*uid*).get()

.addOnSuccessListener **{**

val isProfileDone = **it**.getBoolean(*profileDone*)

if (isProfileDone != null && isProfileDone) {

val intent = Intent(requireContext(), MainActivity::class.*java*)

intent.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*)

requireActivity().finish()

startActivity(intent)

} else {

val map = *mutableMapOf*<String, Any>()

map["login"] = login

map[*profileDone*] = false

FirebaseFirestore.getInstance().collection("users").document(user.*uid*)

.set(map, SetOptions.merge()).addOnSuccessListener **{**

val intent = Intent(requireContext(), SetupProfileActivity::class.*java*)

intent.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*)

requireActivity().finish()

startActivity(intent)

**}**

}

**}**

}

private fun initVars() {

mCallbacksClient = object : PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

override fun onVerificationCompleted(authCredential: PhoneAuthCredential) {

binding.progressBar.*hide*()

Log.d("LoginFragment", "onVerificationCompleted: Success")

signInWithPhone(authCredential)

}

override fun onVerificationFailed(p0: FirebaseException) {

Log.d("LoginFragment", "onVerificationFailed: ${p0.message}")

}

override fun onCodeSent(

s: String,

resendingToken: PhoneAuthProvider.ForceResendingToken

) {

super.onCodeSent(s, resendingToken)

try {

binding.progressBar.*hide*()

} catch (e: Exception) {

}

mResendingTokenClient = resendingToken

val bundle = Bundle()

bundle.putString(*EXTRA\_CODE\_SENT\_PWD*, s)

*findNavController*().navigate(

R.id.*action\_loginFragment\_to\_phoneConfirmationFragment*,

bundle

)

}

}

binding.phoneNumberEditText.addTextChangedListener(object : TextWatcher {

override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {

}

override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {

if (binding.phoneNumberEditText.*text*.toString().*isNotEmpty*()) {

binding.phoneNumberEditText.setBackgroundResource(R.drawable.*back\_edit\_focused*)

binding.textViewPhoneHint.setTextColor(

ContextCompat.getColor(requireContext(), R.color.*colorPrimary*)

)

} else {

binding.phoneNumberEditText.setBackgroundResource(R.drawable.*back\_edit\_error*)

binding.textViewPhoneHint.setTextColor(

ContextCompat.getColor(requireContext(), R.color.*colorError*)

)

}

}

override fun afterTextChanged(s: Editable?) {

}

})

binding.phoneNumberEditText.setOnFocusChangeListener **{** \_, hasFocus **->**

if (hasFocus) {

binding.phoneNumberEditText.setBackgroundResource(R.drawable.*back\_edit\_focused*)

binding.textViewPhoneHint.setTextColor(

ContextCompat.getColor(requireContext(), R.color.*colorPrimary*)

)

} else {

binding.phoneNumberEditText.setBackgroundResource(R.drawable.*back\_edit\_notfocus*)

binding.textViewPhoneHint.setTextColor(

ContextCompat.getColor(requireContext(), R.color.*colorNotFocus*)

)

}

**}**

}

private fun initGoogleSignVars() {

val gso =

GoogleSignInOptions.Builder(GoogleSignInOptions.*DEFAULT\_SIGN\_IN*)

.requestIdToken(getString(R.string.*default\_web\_client\_id*))

.requestEmail()

.requestProfile()

.build()

mGoogleSignInClient = GoogleSignIn.getClient(requireActivity(), gso)

}

}

private fun saveUserProfile() {

val user = FirebaseAuth.getInstance().*currentUser*!!

val profileUpdates = UserProfileChangeRequest.Builder().*let* **{**

**it**.*displayName* = binding.userNameEditText.*text*.toString()

**it**.*photoUri* = Uri.parse(loadedImg)

**it**.build()

**}**

user.updateProfile(profileUpdates).addOnSuccessListener **{**

val map = *mutableMapOf*<String, Any>()

map["uid"] = user.*uid*

map["userName"] = binding.userNameEditText.*text*.toString()

map["userPhoto"] = loadedImg

map["profileDone"] = true

map["userAddress"] = selectedAddress

map["userAddressText"] = binding.userAddressEditText.*text*.toString()

map["phoneTwo"] = binding.additionalPhoneEditText.*text*.toString()

map["friends"] = 0

map["images"] = 0

map["bio"] = binding.bioEditText.*text*.toString()

FirebaseFirestore.getInstance().collection("users").document(user.*uid*)

.set(map, SetOptions.merge()).addOnSuccessListener **{**

val intent = Intent(this, MainActivity::class.*java*)

intent.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*)

finish()

startActivity(intent)

**}**

**}**

}

val tabs = *listOf*(

getString(R.string.*posts*),

getString(R.string.*taxi*),

getString(R.string.*forFarmers*),

getString(R.string.*travel*)

)

binding.tabsHome.*tabMode* = TabLayout.*MODE\_SCROLLABLE*

val adapterChat = HomePagerAdapter(this.requireActivity())

binding.viewPager.*adapter* = adapterChat

TabLayoutMediator(binding.tabsHome, binding.viewPager) **{** tab, position **->**

tab.*text* = tabs[position]

**}**.attach()

private fun initRelated() {

val db = FirebaseFirestore.getInstance()

val query = db.collection("posts")

.whereEqualTo("category", currentPost!!.*category*)

.whereNotEqualTo("id", currentPost!!.*id*)

.orderBy("id", Query.Direction.*DESCENDING*)

.orderBy("date", Query.Direction.*DESCENDING*)

val options: FirestoreRecyclerOptions<ModelPost> =

FirestoreRecyclerOptions.Builder<ModelPost>().setQuery(query, ModelPost::class.*java*)

.build()

adapterPosts = null

adapterPosts = PostsRecyclerViewAdapter(options, this)

binding.recyclerViewRelatedItems.*adapter* = adapterPosts

binding.relatedItemsSwipeRefresh.*isRefreshing* = false

adapterPosts?.startListening()

}

private fun fillUI() {

if (currentPost!!.*active* == -1) {

binding.deactivatedTextView.*show*()

binding.buttonsLinear.*visibility* = View.*GONE*

}

initAuthor()

updateViewCount()

val dateToShow =

getDateToShow(currentPost!!.*date*.toDate()) + ", ${currentPost!!.*addressText*}"

binding.dateTextView.*text* = dateToShow

if (currentPost!!.*images* != null && currentPost!!.*images*.*isNotEmpty*()) {

adapterP =

PagerAdapterPhotoSlider(

currentPost!!.*images*,

this, this, *lifecycle*

)

binding.viewPagerPostDetail.*adapter* = adapterP

connectViewPager(currentPost!!.*images*.size)

binding.defKurjun.*visibility* = View.*GONE*

} else {

binding.defKurjun.*visibility* = View.*VISIBLE*

}

binding.viewCountTextView.*text* = currentPost!!.*views*.toString()

binding.itemNameTextViewItemDetail.*text* = currentPost!!.*name*

binding.itemCostTextViewPostDetail.*text* = if (currentPost!!.*cost* == 0.0)

getString(R.string.*costAgree*)

else

currentPost!!.*costText*

binding.infoItemMagPostDetail.*text* = currentPost!!.*description*

val text = "http://www.helloleylek.kg/$ref"

binding.shareButton.setOnClickListener **{**

val dialog = CustomShare(getString(R.string.*sharePost*), text, this)

dialog.show(*supportFragmentManager*, "shareCustom")

**}**

binding.callButton.setOnClickListener **{**

val dialog =

CustomCallUser(currentPost!!.*phoneOne*, currentPost!!.*phoneTwo*, this)

dialog.show(*supportFragmentManager*, "callUser")

**}**

if (user != null) {

if (user!!.*uid* != currentPost!!.*authorUid*) {

checkIfSaved()

binding.writeButton.setOnClickListener **{**

goToChatWith(user!!, currentPost!!.*authorUid*)

**}**

} else {

binding.progressBar.*hide*()

binding.buttonsLinear.*visibility* = View.*GONE*

if (currentPost!!.*active* != -1)

binding.authorIsMe.*visibility* = View.*VISIBLE*

}

} else {

binding.progressBar.*hide*()

binding.writeButton.setOnClickListener **{**

startActivity(Intent(this, LoginActivity::class.*java*))

**}**

binding.saveButton.setOnClickListener **{**

startActivity(Intent(this, LoginActivity::class.*java*))

**}**

}

}

private fun goToChatWith(user: FirebaseUser, authorUid: String) {

binding.progressBar.*show*()

FirebaseFirestore.getInstance().collection("chats")

.whereArrayContains("users", user.*uid*)

.get().addOnCompleteListener **{**

if (**it**.*isSuccessful*) {

if (**it**.*result* != null && **it**.*result*.*documents*.size > 0) {

var new = true

for (r in **it**.*result*.*documents*) {

val model = r.toObject(ModelChat::class.*java*)

if (model != null && model.*users*.contains(authorUid)) {

new = false

break

}

}

if (new)

createNewChat(user.*uid*, authorUid)

else

goToChat(**it**.*result*.*documents*[0].*reference*.*path*, user.*uid*, authorUid)

} else {

createNewChat(user.*uid*, authorUid)

}

}

**}**

}

private fun createNewChat(uid: String, authorUid: String) {

val chat = ModelChat(

uid + authorUid,

*arrayListOf*(uid, authorUid),

"http://www.helloleylek.kg/$ref",

uid, 1, Timestamp(Calendar.getInstance().*time*),

false

)

val message = ModelMessage()

message.*messageText* = "http://www.helloleylek.kg/$ref"

message.*messageType* = 1

message.*senderUid* = uid

message.*read* = false

FirebaseFirestore.getInstance().collection("chats")

.add(chat).addOnCompleteListener **{** task **->**

if (task.*isSuccessful*) {

task.*result*.collection("messages")

.add(message).addOnSuccessListener **{**

goToChat(task.*result*.*path*, uid, authorUid)

**}**

} else {

binding.progressBar.*hide*()

*toast*(getString(R.string.*tryAgainError*))

}

**}**

}

class LeylekBilimFragment : Fragment(), BilimCatRecyclerViewAdapter.BilimCatRecyclerListener,

AppBarLayout.OnOffsetChangedListener,

BilimCategoriesRecyclerAdapter.BilimCategoriesRecyclerListener {

private var \_binding: FragmentLeylekBilimBinding? = null

private val binding: FragmentLeylekBilimBinding get() = \_binding!!

private var adapter: BilimCatRecyclerViewAdapter? = null

private var adapter2: BilimCatRecyclerViewAdapter? = null

private var adapter3: BilimCategoriesRecyclerAdapter? = null

override fun onCreateView(

inflater: LayoutInflater, container: ViewGroup?,

savedInstanceState: Bundle?

): View {

\_binding = FragmentLeylekBilimBinding.inflate(inflater, container, false)

return binding.*root*

}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

super.onViewCreated(view, savedInstanceState)

initUserName()

initRecyclerView1()

binding.appbar.addOnOffsetChangedListener(this)

}

private fun initUserName() {

val user = FirebaseAuth.getInstance().*currentUser*

if (user != null) {

FirebaseFirestore.getInstance().collection("users")

.document(user.*uid*).get().addOnSuccessListener **{**

val userName = **it**.getString("userName")

val text = "${getString(R.string.*salam*)} $userName,"

binding.salamTextView.*text* = text

**}**

}

}

override fun onResume() {

super.onResume()

if (adapter == null)

initRecyclerView1()

else

adapter?.startListening()

if (adapter2 == null)

initRecyclerView2()

else

adapter2?.startListening()

if (adapter3 == null)

initRecyclerView3()

else

adapter3?.startListening()

requireActivity().*window*.*statusBarColor* =

ContextCompat.getColor(requireContext(), R.color.*backBilimTop*)

}

override fun onStop() {

super.onStop()

adapter?.stopListening()

adapter2?.stopListening()

adapter3?.stopListening()

}

private fun initRecyclerView1() {

binding.progressBar.*show*()

val query = FirebaseFirestore.getInstance().collection("bilim")

.orderBy("views", Query.Direction.*DESCENDING*)

.limit(5)

val options: FirestoreRecyclerOptions<ModelBilimCourse> =

FirestoreRecyclerOptions.Builder<ModelBilimCourse>()

.setQuery(query, ModelBilimCourse::class.*java*).build()

adapter = BilimCatRecyclerViewAdapter(options, this, 1)

binding.recyclerViewMostPopular.*adapter* = adapter

adapter?.startListening()

initRecyclerView2()

}

private fun initRecyclerView2() {

val query = FirebaseFirestore.getInstance().collection("bilim")

.orderBy("date", Query.Direction.*DESCENDING*)

.limit(5)

val options: FirestoreRecyclerOptions<ModelBilimCourse> =

FirestoreRecyclerOptions.Builder<ModelBilimCourse>()

.setQuery(query, ModelBilimCourse::class.*java*).build()

adapter2 = BilimCatRecyclerViewAdapter(options, this, 2)

binding.recyclerViewNewest.*adapter* = adapter2

adapter2?.startListening()

initRecyclerView3()

}

private fun initRecyclerView3() {

val query = FirebaseFirestore.getInstance().collection("bilimcats")

.orderBy("id", Query.Direction.*ASCENDING*)

val options: FirestoreRecyclerOptions<ModelBilimCategory> =

FirestoreRecyclerOptions.Builder<ModelBilimCategory>()

.setQuery(query, ModelBilimCategory::class.*java*).build()

adapter3 = BilimCategoriesRecyclerAdapter(options, this)

binding.recyclerViewCategories.*adapter* = adapter3

adapter3?.startListening()

binding.progressBar.*hide*()

}

override fun onCategoryClick(current: ModelBilimCategory) {

Intent(requireContext(), CoursesByCategoryActivity::class.*java*).*let* **{** intent **->**

intent.putExtra(*EXTRA\_BILIM\_CAT*, current)

startActivity(intent)

**}**

}

override fun onBilimCatClick(position: Int, code: Int) {

val ref = if (code == 1)

adapter!!.*snapshots*.getSnapshot(position).*reference*.*path*

else

adapter2!!.*snapshots*.getSnapshot(position).*reference*.*path*

Intent(requireContext(), BilimCourseActivity::class.*java*).*let* **{** intent **->**

intent.putExtra(*EXTRA\_BILIM\_CAT*, ref)

startActivity(intent)

**}**

}

override fun onOffsetChanged(appBarLayout: AppBarLayout?, verticalOffset: Int) {

if (*abs*(verticalOffset) >= (appBarLayout!!.*totalScrollRange*)) {

requireActivity().*window*.*statusBarColor* = Color.*WHITE*

} else {

requireActivity().*window*.*statusBarColor* =

ContextCompat.getColor(requireContext(), R.color.*backBilimTop*)

}

}

}

class BilimCourseActivity : AppCompatActivity(), CustomCallUser.CallUserListener,

CustomShare.CustomShareListener, CustomMenuBilimUser.CustomMenuBilimListener,

AppBarLayout.OnOffsetChangedListener, CustomWriteOrReview.CustomWriteOrReviewListener {

private lateinit var binding: ActivityBilimCourseBinding

private lateinit var bilimCourse: ModelBilimCourse

private var user: FirebaseUser? = null

private var ref: String = ""

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

binding = ActivityBilimCourseBinding.inflate(*layoutInflater*)

setContentView(binding.*root*)

*window*.*statusBarColor* = ContextCompat.getColor(this, R.color.*white4*)

user = FirebaseAuth.getInstance().*currentUser*

binding.progressBar.*show*()

if (*intent*.hasExtra(*EXTRA\_BILIM\_CAT*)) {

ref = *intent*.getStringExtra(*EXTRA\_BILIM\_CAT*)!!

getCourse()

} else {

getRefFromUri()

}

binding.appBar.addOnOffsetChangedListener(this)

binding.arrBack.setOnClickListener **{**

onBackPressed()

**}**

binding.arrBack2.setOnClickListener **{**

onBackPressed()

**}**

}

private fun getRefFromUri() {

val uri = *intent*.*data*.*toString*()

var last = uri.*lastIndex*

var id = ""

while (uri[last] != '/') {

id += uri[last]

last--

}

val postId = id.*reversed*()

ref = "bilim/$postId"

getCourse()

}

private fun getCourse() {

FirebaseFirestore.getInstance().document(ref).addSnapshotListener **{** value, error **->**

if (error != null) {

*toast*(getString(R.string.*tryAgainError*))

binding.progressBar.*hide*()

goToMain()

}

if (value != null && value.exists()) {

val temp = value.toObject(ModelBilimCourse::class.*java*)

if (temp != null) {

bilimCourse = temp

checkIfUserBoughtThisCourse()

} else {

*toast*(getString(R.string.*courseNotFound*))

binding.progressBar.*hide*()

goToMain()

}

} else {

*toast*(getString(R.string.*courseNotFound*))

binding.progressBar.*hide*()

goToMain()

}

**}**

}

private fun checkIfUserBoughtThisCourse() {

val user = FirebaseAuth.getInstance().*currentUser*

if (user == null) {

initUI(0)

} else {

FirebaseFirestore.getInstance().document(ref).collection("users")

.document(user.*uid*).get().addOnSuccessListener **{**

Log.d("NURIKO", "checkIfUserBoughtThisCourse ${user.*uid*}")

if (**it**.exists()) {

initUI(1)

} else {

initUI(0)

}

**}**.addOnFailureListener **{**

initUI(0)

**}**

}

}

private fun goToMain() {

Intent(this, MainActivity::class.*java*).*let* **{** intent **->**

intent.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*)

finish()

startActivity(intent)

**}**

}

private fun initUI(isUserBoughtCourse: Int) {

val cost: Double = bilimCourse.*cost* ?: 0.0

val edit = getSharedPreferences(*BILIM\_CAT*, Context.*MODE\_PRIVATE*).edit()

edit.putString(*CURRENT\_BILIM\_CAT*, ref)

edit.putFloat(*CURRENT\_BILIM\_COST*, cost.toFloat())

edit.putInt(*VIDEO\_COUNT\_KEY*, bilimCourse.*videoCount*)

edit.putInt(*TEST\_COUNT\_KEY*, bilimCourse.*testCount*)

edit.putInt(*ADDMAT\_COUNT\_KEY*, bilimCourse.*addMatCount*)

edit.putInt(*IS\_USER\_BOUGHT\_KEY*, isUserBoughtCourse)

edit.apply()

val tabs = *listOf*(

getString(R.string.*videoLessons*),

getString(R.string.*tests*),

getString(R.string.*additionalMat*)

)

binding.tabs.*tabMode* = TabLayout.*MODE\_SCROLLABLE*

val adapterChat = BilimCatPagerAdapter(this)

binding.viewPager.*adapter* = adapterChat

TabLayoutMediator(binding.tabs, binding.viewPager) **{** tab, position **->**

tab.*text* = tabs[position]

**}**.attach()

initOwner(bilimCourse.*ownerUid*)

Glide.with(this).load(bilimCourse.*icon*)

.error(ContextCompat.getDrawable(this, R.drawable.*jrt1*))

.into(binding.courseIcon)

binding.courseName.*text* = bilimCourse.*name*

binding.courseName2.*text* = bilimCourse.*name*

binding.videoCountTextView.*text* = bilimCourse.*videoCount*.toString()

binding.testCountTextView.*text* = bilimCourse.*testCount*.toString()

val ratingStr = bilimCourse.*rating*.toString()

binding.reviewCountTextView.*text* = if (ratingStr.length > 4)

ratingStr.*take*(4)

else

ratingStr

binding.likeCountTextView.*text* = bilimCourse.*likes*.toString()

binding.viewCountTextView.*text* = bilimCourse.*views*.toString()

if (user != null) {

if (user!!.*uid* == bilimCourse.*ownerUid*) {

binding.writeButton.*visibility* = View.*GONE*

binding.likeButton.*visibility* = View.*GONE*

} else if (binding.likeButton.*visibility* == View.*VISIBLE*) {

checkIfLiked(user!!.*uid*)

}

} else {

binding.writeButton.*visibility* = View.*GONE*

binding.likeButton.*visibility* = View.*GONE*

}

binding.likeButton.setOnClickListener **{**

binding.likeButton.*visibility* = View.*GONE*

likeCourse()

**}**

binding.infoButton.setOnClickListener **{**

val dialog = CustomInfoView(bilimCourse.*description*)

dialog.show(*supportFragmentManager*, "courseInfo")

**}**

binding.writeButton.setOnClickListener **{**

val dialog = CustomWriteOrReview(this)

dialog.show(*supportFragmentManager*, "writeOrReview")

**}**

binding.menuButton.setOnClickListener **{**

val dialog = CustomMenuBilimUser(this, bilimCourse.*ownerUid*)

dialog.show(*supportFragmentManager*, "showMenuBilimCat")

**}**

}

private fun likeCourse() {

val map1 = *mutableMapOf*<String, Any>()

map1["user"] = user!!.*uid*

val map2 = *mutableMapOf*<String, Any>()

map2["course"] = bilimCourse.*id*

val map3 = *mutableMapOf*<String, Any>()

map3["likes"] = bilimCourse.*likes* + 1

val db = FirebaseFirestore.getInstance()

db.document(ref)

.collection("likes")

.document(user!!.*uid*).set(map1).addOnSuccessListener **{**

db.collection("users")

.document(user!!.*uid*).collection("likesCourses")

.document(bilimCourse.*id*).set(map2)

.addOnSuccessListener **{**

db.document(ref).set(map3, SetOptions.merge()).addOnSuccessListener **{**

*toast*(getString(R.string.*thanksForLike*))

**}**

**}**

**}**.addOnFailureListener **{**

*toast*(getString(R.string.*tryAgainError*))

binding.likeButton.*visibility* = View.*VISIBLE*

**}**

}

private fun checkIfLiked(uid: String) {

FirebaseFirestore.getInstance().document(ref)

.collection("likes").document(uid)

.get().addOnSuccessListener **{**

if (**it**.exists())

binding.likeButton.*visibility* = View.*GONE*

else

binding.likeButton.*visibility* = View.*VISIBLE*

**}**.addOnFailureListener **{**

Log.d("NURIKO", "checkIfLiked: $**it**")

**}**

}

private fun initOwner(ownerUid: String) {

FirebaseFirestore.getInstance().collection("users").document(ownerUid)

.get().addOnCompleteListener **{**

if (**it**.*isSuccessful* && **it**.*result*.exists()) {

val userName = **it**.*result*.getString("userName")

binding.teacherNameTextView.*text* = userName

} else {

binding.teacherNameTextView.*text* = "-"

}

binding.progressBar.*hide*()

**}**

}

override fun onCallClick() {

if (bilimCourse.*phoneNumbers* != null) {

val phone1 = bilimCourse.*phoneNumbers*.*getOrNull*(0)

val phone2 = bilimCourse.*phoneNumbers*.*getOrNull*(1)

val dialog = CustomCallUser(phone1, phone2, this, "fafa")

dialog.show(*supportFragmentManager*, "callUser")

}

}

override fun onShareClick() {

val link = "http://www.helloleylek.kg/bilim/${bilimCourse.*id*}"

val dialog = CustomShare(getString(R.string.*shareBilimCourse*), link, this)

dialog.show(*supportFragmentManager*, "shareBilimOwner")

}

override fun onShowProfileClick() {

Intent(this, UserProfileActivity::class.*java*).*let* **{** intent **->**

intent.putExtra(*EXTRA\_AL\_UID*, bilimCourse.*ownerUid*)

startActivity(intent)

**}**

}

override fun shareSocial(link: String) {

ShareCompat.IntentBuilder.from(this)

.setType("text/plain")

.setChooserTitle(getString(R.string.*shareWith2*))

.setText(link)

.startChooser()

}

private fun goToChatWith(user: FirebaseUser, authorUid: String) {

if (user.*uid* != authorUid) {

binding.progressBar.*show*()

FirebaseFirestore.getInstance().collection("chats")

.whereArrayContains("users", user.*uid*)

.get().addOnCompleteListener **{**

if (**it**.*isSuccessful*) {

if (**it**.*result* != null && **it**.*result*.*documents*.size > 0) {

var new = true

for (r in **it**.*result*.*documents*) {

val model = r.toObject(ModelChat::class.*java*)

if (model != null && model.*users*.contains(authorUid)) {

new = false

break

}

}

if (new)

createNewChat(user.*uid*, authorUid)

else

sendInterestedMessage(

**it**.*result*.*documents*[0].*reference*.*path*,

user.*uid*,

authorUid

)

} else {

createNewChat(user.*uid*, authorUid)

}

}

**}**

} else {

*toast*("you can't write to yourself")

}

}

private fun sendInterestedMessage(path: String, uid: String, authorUid: String) {

val message = ModelMessage()

message.*messageText* = "http://www.helloleylek.kg/bilim/${bilimCourse.*id*}"

message.*messageType* = 1

message.*senderUid* = uid

message.*read* = false

FirebaseFirestore.getInstance().document(path)

.collection("messages")

.add(message).addOnSuccessListener **{**

goToChat(**it**.*path*, uid, authorUid)

**}**

}

private fun createNewChat(uid: String, authorUid: String) {

val chat = ModelChat(

uid + authorUid,

*arrayListOf*(uid, authorUid),

"http://www.helloleylek.kg/bilim/${bilimCourse.*id*}",

uid, 1, Timestamp(Calendar.getInstance().*time*),

false

)

val message = ModelMessage()

message.*messageText* = "http://www.helloleylek.kg/bilim/${bilimCourse.*id*}"

message.*messageType* = 1

message.*senderUid* = uid

message.*read* = false

FirebaseFirestore.getInstance().collection("chats")

.add(chat).addOnCompleteListener **{** task **->**

if (task.*isSuccessful*) {

task.*result*.collection("messages")

.add(message).addOnSuccessListener **{**

goToChat(task.*result*.*path*, uid, authorUid)

**}**

} else {

binding.progressBar.*hide*()

*toast*(getString(R.string.*tryAgainError*))

}

**}**

}

private fun goToChat(path: String, menUid: String, alUid: String) {

binding.progressBar.*hide*()

Intent(this, PrivateChatActivity::class.*java*).*let* **{**

**it**.putExtra(*EXTRA\_MEN\_UID*, menUid)

**it**.putExtra(*EXTRA\_AL\_UID*, alUid)

**it**.putExtra(*EXTRA\_CHAT\_REF*, path)

startActivity(**it**)

**}**

}

override fun callUserPhone(phone: String) {

val callIntent = Intent(Intent.*ACTION\_DIAL*)

callIntent.*data* = Uri.parse("tel:$phone")

startActivity(callIntent)

}

override fun onOffsetChanged(appBarLayout: AppBarLayout?, verticalOffset: Int) {

if (*abs*(verticalOffset) >= (appBarLayout!!.*totalScrollRange*)) {

*window*.*statusBarColor* = Color.*WHITE*

binding.coordinatorLayout.setBackgroundColor(Color.*WHITE*)

binding.appBar2.*show*()

} else {

binding.coordinatorLayout.setBackgroundColor(

ContextCompat.getColor(this, R.color.*backBilimTop*)

)

binding.appBar2.*hide*()

*window*.*statusBarColor* = ContextCompat.getColor(this, R.color.*white4*)

}

}

override fun onWriteReviewSelected() {

Intent(this, WriteCourseReviewActivity::class.*java*).*let* **{** intent **->**

intent.putExtra(*EXTRA\_COURSE\_ID*, bilimCourse.*id*)

intent.putExtra(*EXTRA\_CURRENT\_RATING*, bilimCourse.*rating*)

startActivity(intent)

**}**

}

override fun onWriteToOwnerSelected() {

val user = FirebaseAuth.getInstance().*currentUser*

if (user == null) {

*toast*(getString(R.string.*pleaseRegister*))

}else {

goToChatWith(user, bilimCourse.*ownerUid*)

}

}

}

class VideoLessonsFragment : Fragment(),

VideoLessonRecyclerViewAdapter.VideoLessonRecyclerListener,

CustomBuyCourse.CustomBuyClickListener, CustomCallAdmin.CustomCallAdminListener {

private var \_binding: FragmentVideoLessonsBinding? = null

private val binding: FragmentVideoLessonsBinding get() = \_binding!!

private var adapter: VideoLessonRecyclerViewAdapter? = null

private var currentCourseRef: String = ""

private var currentCourseCost: Float = 0F

private var videoCount: Int = 0

private var testCount: Int = 0

private var addMatCount: Int = 0

private var isUserBought: Int = 0

private var currentCourseId: String = ""

private lateinit var phones: ModelPhone

private var user: FirebaseUser? = null

override fun onCreateView(

inflater: LayoutInflater, container: ViewGroup?,

savedInstanceState: Bundle?

): View {

\_binding = FragmentVideoLessonsBinding.inflate(inflater, container, false)

return binding.*root*

}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

super.onViewCreated(view, savedInstanceState)

binding.progressBar.*show*()

user = FirebaseAuth.getInstance().*currentUser*

val sp = requireActivity().getSharedPreferences(*BILIM\_CAT*, *MODE\_PRIVATE*)

currentCourseRef = sp.getString(*CURRENT\_BILIM\_CAT*, "") ?: ""

currentCourseCost = sp.getFloat(*CURRENT\_BILIM\_COST*, 0f)

videoCount = sp.getInt(*VIDEO\_COUNT\_KEY*, 0)

testCount = sp.getInt(*TEST\_COUNT\_KEY*, 0)

addMatCount = sp.getInt(*ADDMAT\_COUNT\_KEY*, 0)

isUserBought = sp.getInt(*IS\_USER\_BOUGHT\_KEY*, 0)

initCurrentCourseId()

initPhones()

if (currentCourseRef != "")

initRecyclerView(currentCourseRef)

else

requireActivity().*toast*("course id: 0 $currentCourseRef")

binding.swipeRefresh.setOnRefreshListener **{**

if (currentCourseRef != "") {

if (adapter == null) {

initRecyclerView(currentCourseRef)

} else {

adapter?.startListening()

binding.swipeRefresh.*isRefreshing* = false

}

}

**}**

}

private fun initCurrentCourseId() {

var temp = ""

var i = currentCourseRef.*lastIndex*

while (currentCourseRef[i] != '/') {

temp += currentCourseRef[i]

i--

}

currentCourseId = temp

}

private fun initPhones() {

FirebaseFirestore.getInstance().collection("admin")

.document("phones").get()

.addOnSuccessListener **{**

phones = **it**.toObject(ModelPhone::class.*java*)!!

**}**

}

override fun onResume() {

super.onResume()

if (currentCourseRef != "") {

if (adapter == null) {

initRecyclerView(currentCourseRef)

} else {

adapter?.startListening()

binding.swipeRefresh.*isRefreshing* = false

}

} else {

requireActivity().*toast*("Author id: 0 $currentCourseRef")

}

}

override fun onStop() {

super.onStop()

adapter?.stopListening()

}

private fun initRecyclerView(courseRef: String) {

val query = FirebaseFirestore.getInstance()

.document(courseRef).collection("video")

.orderBy("order", Query.Direction.*ASCENDING*)

val options: FirestoreRecyclerOptions<ModelVideoLesson> =

FirestoreRecyclerOptions.Builder<ModelVideoLesson>()

.setQuery(query, ModelVideoLesson::class.*java*).build()

adapter = VideoLessonRecyclerViewAdapter(options, this, isUserBought)

binding.recyclerView.*adapter* = adapter

adapter?.startListening()

binding.swipeRefresh.*isRefreshing* = false

binding.progressBar.*hide*()

}

override fun onVideoClick(position: Int, isFree: Boolean) {

val snapshot = adapter!!.*snapshots*.getSnapshot(position)

val link = snapshot.getString("videoUrl")

val name = snapshot.getString("name")

val ref = snapshot.*reference*.*path*

val user = FirebaseAuth.getInstance().*currentUser*

if (user != null) {

if (isFree) {

freeCourse(link, name, ref)

} else {

buyCourse()

}

} else {

Intent(requireContext(), LoginActivity::class.*java*).*let* **{** intent **->**

intent.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*)

requireActivity().finish()

startActivity(intent)

**}**

}

}

private fun buyCourse() {

val text =

"${getString(R.string.*buyCourseCostInfo*)} \n\n\t- $videoCount ${getString(R.string.*videos*)}\n\t- $testCount ${

getString(R.string.*tests*)

}\n\t- $addMatCount ${getString(R.string.*additionalMat*)} "

val dialog = CustomBuyCourse(text, currentCourseCost, this)

dialog.show(requireActivity().*supportFragmentManager*, "BuyCourseView")

}

private fun freeCourse(link: String?, name: String?, ref: String) {

Intent(requireContext(), VideoPlayerFullActivity::class.*java*).*let* **{** intent **->**

intent.putExtra(*EXTRA\_VIDEO\_URL*, link)

intent.putExtra(*EXTRA\_VIDEO\_lESSON\_NAME*, name)

intent.putExtra(*EXTRA\_VIDEO\_LESSON\_REF*, ref)

startActivity(intent)

**}**

}

override fun payWithCard() {

Toast.makeText(requireContext(), getString(R.string.*comingSoon*), Toast.*LENGTH\_SHORT*).show()

}

private fun onPaymentTypeSelected(type: Int) {

if (user != null) {

binding.progressBar.*show*()

val map = *mutableMapOf*<String, Any>()

map["courseRef"] = currentCourseRef

map["user"] = user!!.*uid*

map["cost"] = currentCourseCost

FirebaseFirestore.getInstance().collection("courseERequests")

.document(currentCourseId + user!!.*uid*)

.set(map).addOnSuccessListener **{**

binding.progressBar.*hide*()

Intent(requireContext(), PayWithEWalletActivity::class.*java*).*let* **{**

**it**.putExtra(*EXTRA\_COURSE\_COST*, currentCourseCost)

**it**.putExtra(*EXTRA\_PHONE*, phones.*phoneM*)

**it**.putExtra(*EXTRA\_PAYMENT\_TYPE*, type)

startActivity(**it**)

**}**

**}**

} else {

Intent(requireContext(), LoginActivity::class.*java*).*let* **{** intent **->**

intent.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*)

requireActivity().finish()

startActivity(intent)

**}**

}

}

override fun payWithMegacom() {

onPaymentTypeSelected(3)

}

override fun payWithBeeline() {

onPaymentTypeSelected(2)

}

override fun payWithO() {

onPaymentTypeSelected(1)

}

override fun payCash() {

onPaymentTypeSelected(0)

}

override fun callAdmin() {

val dialog = CustomCallAdmin(phones, this)

dialog.show(requireActivity().*supportFragmentManager*, "callAdmin")

}

override fun onChatWhatsApp(whatsAppPhone: String) {

val url = "https://api.whatsapp.com/send?phone=$whatsAppPhone"

try {

val pm = requireContext().*packageManager*

pm.getPackageInfo("com.whatsapp", PackageManager.*GET\_ACTIVITIES*)

val i = Intent(Intent.*ACTION\_VIEW*)

i.*data* = Uri.parse(url)

startActivity(i)

} catch (e: PackageManager.NameNotFoundException) {

requireActivity().*toast*(getString(R.string.*whatsAppNotInstalled*))

e.printStackTrace()

}

}

override fun onCallPhone(phones: String) {

val callIntent = Intent(Intent.*ACTION\_DIAL*)

callIntent.*data* = Uri.parse("tel:$phones")

startActivity(callIntent)

}

override fun onChatHere() {

binding.progressBar.*show*()

if (user != null) {

val db = FirebaseFirestore.getInstance()

db.collection("chats").document(user!!.*uid* + "admin")

.get().addOnSuccessListener **{**

if (**it**.exists()) {

goToChat(**it**.*reference*.*path*, user!!.*uid*)

} else {

createNewChat(user!!.*uid*)

}

**}**

}

}

private fun createNewChat(uid: String) {

val authorUid = "admin"

val chat = ModelChat(

uid + authorUid,

*arrayListOf*(uid, authorUid),

getString(R.string.*chatCreated*),

uid, 1, Timestamp(Calendar.getInstance().*time*),

false

)

val message = ModelMessage()

message.*messageText* = getString(R.string.*chatCreated*)

message.*messageType* = 1

message.*senderUid* = uid

message.*read* = false

FirebaseFirestore.getInstance().collection("chats")

.add(chat).addOnCompleteListener **{** task **->**

if (task.*isSuccessful*) {

task.*result*.collection("messages")

.add(message).addOnSuccessListener **{**

goToChat(task.*result*.*path*, uid)

**}**

} else {

binding.progressBar.*hide*()

requireActivity().*toast*(getString(R.string.*tryAgainError*))

}

**}**

}

private fun goToChat(path: String, uid: String) {

binding.progressBar.*hide*()

Intent(requireContext(), PrivateChatActivity::class.*java*).*let* **{**

**it**.putExtra(*EXTRA\_MEN\_UID*, uid)

**it**.putExtra(*EXTRA\_AL\_UID*, "admin")

**it**.putExtra(*EXTRA\_CHAT\_REF*, path)

startActivity(**it**)

**}**

}

}

class TestsFragment : Fragment(), TestRecyclerViewAdapter.TestRecyclerClickListener {

private var \_binding: FragmentTestsBinding? = null

private val binding: FragmentTestsBinding get() = \_binding!!

private var adapter: TestRecyclerViewAdapter? = null

private var currentCourseRef: String = ""

override fun onCreateView(

inflater: LayoutInflater, container: ViewGroup?,

savedInstanceState: Bundle?

): View {

\_binding = FragmentTestsBinding.inflate(inflater, container, false)

return binding.*root*

}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

super.onViewCreated(view, savedInstanceState)

binding.progressBar.*show*()

val sp = requireActivity().getSharedPreferences(*BILIM\_CAT*, Context.*MODE\_PRIVATE*)

currentCourseRef = sp.getString(*CURRENT\_BILIM\_CAT*, "") ?: ""

if (currentCourseRef != "")

initRecyclerView()

else

requireActivity().*toast*("Author id: 0 $currentCourseRef")

binding.swipeRefresh.setOnRefreshListener **{**

if (currentCourseRef != "") {

if (adapter == null) {

initRecyclerView()

} else {

adapter?.startListening()

binding.swipeRefresh.*isRefreshing* = false

}

}

**}**

}

override fun onResume() {

super.onResume()

if (currentCourseRef != "") {

if (adapter == null) {

initRecyclerView()

} else {

adapter?.startListening()

binding.swipeRefresh.*isRefreshing* = false

}

} else {

requireActivity().*toast*("Author id: 0 $currentCourseRef")

}

}

override fun onStop() {

super.onStop()

adapter?.stopListening()

}

private fun initRecyclerView() {

val query = FirebaseFirestore.getInstance().document(currentCourseRef).collection("tests")

.orderBy("id", Query.Direction.*ASCENDING*)

val options: FirestoreRecyclerOptions<ModelTest> =

FirestoreRecyclerOptions.Builder<ModelTest>().setQuery(query, ModelTest::class.*java*)

.build()

adapter = TestRecyclerViewAdapter(options, this)

binding.recyclerView.*adapter* = adapter

adapter?.startListening()

binding.progressBar.*hide*()

binding.swipeRefresh.*isRefreshing* = false

}

override fun onTestClick(position: Int) {

val user = FirebaseAuth.getInstance().*currentUser*

if (user != null) {

val snapshot = adapter!!.*snapshots*.getSnapshot(position)

val ref = snapshot.*reference*.*path*

val name = snapshot.getString("name")

Intent(requireContext(), TestActualActivity::class.*java*).*let* **{** intent **->**

intent.putExtra(*EXTRA\_TEST\_REF*, ref)

intent.putExtra(*EXTRA\_TEST\_NAME*, name)

startActivity(intent)

**}**

} else {

startActivity(Intent(requireContext(), LoginActivity::class.*java*))

}

}

}

class TestActualActivity : AppCompatActivity(), CustomShowResult.CustomShowResultListener {

private lateinit var binding: ActivityTestActualBinding

private val questions = *mutableListOf*<ModelQuestion>()

private var dots: MutableList<TextView> = *mutableListOf*()

private var ref: String = ""

private var index = 0

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

binding = ActivityTestActualBinding.inflate(*layoutInflater*)

setContentView(binding.*root*)

*window*.*statusBarColor* = ContextCompat.getColor(this, R.color.*purpleBack*)

ref = *intent*.getStringExtra(*EXTRA\_TEST\_REF*) ?: ""

val name = *intent*.getStringExtra(*EXTRA\_TEST\_NAME*)

binding.progressBar.*show*()

binding.toolbar.*title* = name

initUserPhoto()

initQuestions()

}

private fun initViewQuestions() {

if (questions.*isNotEmpty*()) {

initUI()

binding.buttonNext.setOnClickListener **{**

if (index < questions.size - 1) {

index++

initUI()

} else {

val dialog = CustomShowResult(this)

dialog.show(*supportFragmentManager*, "showResults")

}

**}**

binding.buttonPrev.setOnClickListener **{**

if (index > 0) {

index--

initUI()

}

**}**

binding.buttonA.setOnClickListener **{**

chooseA()

questions[index].userAnswer = 1

removeB()

removeC()

removeD()

removeE()

**}**

binding.buttonB.setOnClickListener **{**

chooseB()

questions[index].userAnswer = 2

removeA()

removeC()

removeD()

removeE()

**}**

binding.buttonC.setOnClickListener **{**

chooseC()

questions[index].userAnswer = 3

removeB()

removeA()

removeD()

removeE()

**}**

binding.buttonD.setOnClickListener **{**

chooseD()

questions[index].userAnswer = 4

removeB()

removeC()

removeA()

removeE()

**}**

binding.buttonE.setOnClickListener **{**

chooseE()

questions[index].userAnswer = 5

removeB()

removeC()

removeD()

removeA()

**}**

}

}

private fun initUI() {

removeA()

removeB()

removeC()

removeD()

removeE()

if (questions.size > 1)

createDots(index, questions.size)

if (index == 0)

binding.buttonPrev.*visibility* = View.*GONE*

else

binding.buttonPrev.*visibility* = View.*VISIBLE*

if (index == questions.size - 1)

binding.buttonNext.*text* = getString(R.string.*finishTest*)

else

binding.buttonNext.*text* = getString(R.string.*next*)

binding.questionTextView.*text* = questions[index].question

binding.variantATextView.*text* = questions[index].varA

binding.variantBTextView.*text* = questions[index].varB

binding.variantCTextView.*text* = questions[index].varC

binding.variantDTextView.*text* = questions[index].varD

if (questions[index].varE != null) {

binding.variantETextView.*text* = questions[index].varE

binding.variantETextView.*show*()

} else {

binding.variantETextView.*hide*()

}

if (questions[index].photo.*isNullOrEmpty*())

binding.imageCardView.*visibility* = View.*GONE*

else {

binding.imageCardView.*visibility* = View.*VISIBLE*

Glide.with(this).load(questions[index].photo)

.error(ContextCompat.getDrawable(this, R.drawable.*def*))

.into(binding.imageView)

}

if (questions[index].userAnswer != -1 && questions[index].userAnswer != null)

alreadyAnswered(questions[index].userAnswer!!)

}

private fun alreadyAnswered(userAnswer: Int) {

when (userAnswer) {

1 -> {

chooseA()

}

2 -> {

chooseB()

}

3 -> {

chooseC()

}

4 -> {

chooseD()

}

5 -> {

chooseE()

}

}

}

private fun chooseA() {

binding.buttonA.setBackgroundResource(R.drawable.*back\_question\_a*)

binding.variantATextView

.setTextColor(ContextCompat.getColor(this, R.color.*colorWhite*))

}

private fun chooseB() {

binding.buttonB.setBackgroundResource(R.drawable.*back\_question\_a*)

binding.variantBTextView

.setTextColor(ContextCompat.getColor(this, R.color.*colorWhite*))

}

private fun chooseC() {

binding.buttonC.setBackgroundResource(R.drawable.*back\_question\_a*)

binding.variantCTextView

.setTextColor(ContextCompat.getColor(this, R.color.*colorWhite*))

}

private fun chooseD() {

binding.buttonD.setBackgroundResource(R.drawable.*back\_question\_a*)

binding.variantDTextView

.setTextColor(ContextCompat.getColor(this, R.color.*colorWhite*))

}

private fun chooseE() {

binding.buttonE.setBackgroundResource(R.drawable.*back\_question\_a*)

binding.variantETextView

.setTextColor(ContextCompat.getColor(this, R.color.*colorWhite*))

}

private fun removeA() {

binding.buttonA.setBackgroundResource(R.drawable.*back\_question\_na*)

binding.variantATextView

.setTextColor(ContextCompat.getColor(this, R.color.*textColorDark*))

}

private fun removeB() {

binding.buttonB.setBackgroundResource(R.drawable.*back\_question\_na*)

binding.variantBTextView

.setTextColor(ContextCompat.getColor(this, R.color.*textColorDark*))

}

private fun removeC() {

binding.buttonC.setBackgroundResource(R.drawable.*back\_question\_na*)

binding.variantCTextView

.setTextColor(ContextCompat.getColor(this, R.color.*textColorDark*))

}

private fun removeD() {

binding.buttonD.setBackgroundResource(R.drawable.*back\_question\_na*)

binding.variantDTextView

.setTextColor(ContextCompat.getColor(this, R.color.*textColorDark*))

}

private fun removeE() {

binding.buttonE.setBackgroundResource(R.drawable.*back\_question\_na*)

binding.variantETextView

.setTextColor(ContextCompat.getColor(this, R.color.*textColorDark*))

}

private fun initQuestions() {

FirebaseFirestore.getInstance().document(ref)

.collection("questions").get()

.addOnCompleteListener **{**

if (**it**.*isSuccessful*) {

if (!**it**.*result*.*isEmpty*) {

for (sn in **it**.*result*) {

val question = sn.getString("question")

val varA = sn.getString("varA")

val varB = sn.getString("varB")

val varC = sn.getString("varC")

val varD = sn.getString("varD")

val varE = sn.getString("varE")

val answer = sn.getLong("answer")!!.toInt()

val description = sn.getString("description")

val photo = sn.getString("photo")

val model = ModelQuestion(

question!!, varA!!, varB!!,

varC!!, varD!!, varE,

answer, description, photo

)

model.userAnswer = -1

questions.add(model)

}

binding.progressBar.*hide*()

initViewQuestions()

} else {

*toast*("NO Questions! Empty")

binding.progressBar.*hide*()

}

} else {

*toast*("ERROR ${**it**.*exception*}")

binding.progressBar.*hide*()

}

**}**

}

private fun createDots(position: Int, size: Int) {

binding.linearLayoutForDots.removeAllViews()

var i = 0

while (i < size) {

dots.add(TextView(this))

if (i == position) {

dots[i].setBackgroundResource(

R.drawable.*back\_count*

)

dots[i].setTextColor(

ContextCompat.getColor(this, R.color.*colorWhite*)

)

} else {

dots[i].setTextColor(

ContextCompat.getColor(this, R.color.*textColorDark*)

)

dots[i].setBackgroundResource(R.drawable.*back\_count\_uns*)

}

dots[i].*text* = (i + 1).toString()

dots[i].*textAlignment* = TextView.*TEXT\_ALIGNMENT\_CENTER*

val params = LinearLayout.LayoutParams(

ViewGroup.LayoutParams.*WRAP\_CONTENT*,

ViewGroup.LayoutParams.*WRAP\_CONTENT*

)

dots[i].*rightPaddingDp* = 5F

dots[i].*leftPaddingDp* = 5F

params.setMargins(12, 0, 12, 0)

binding.linearLayoutForDots.addView(dots[i], params)

i++

}

}

private fun initUserPhoto() {

val user = FirebaseAuth.getInstance().*currentUser*!!

FirebaseFirestore.getInstance().collection("users")

.document(user.*uid*).get().addOnCompleteListener **{**

if (**it**.*isSuccessful*) {

val userPhoto = **it**.*result*.getString("userPhoto")

if (!userPhoto.*isNullOrEmpty*())

Glide.with(this).load(userPhoto)

.into(binding.userAvatar)

}

**}**

}

override fun onShowResultConfirmed() {

val modelQuestions = ModelQuestions(questions)

Intent(this, ResultsActivity::class.*java*).*let* **{** intent **->**

intent.putExtra(*EXTRA\_QUESTIONS*, modelQuestions)

finish()

startActivity(intent)

**}**

}

}

class ResultsActivity : AppCompatActivity() {

private lateinit var binding: ActivityResultsBinding

private lateinit var questions: List<ModelQuestion>

private lateinit var adapter: ResultsRecyclerViewAdapter

private var score: Int = 0

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

binding = ActivityResultsBinding.inflate(*layoutInflater*)

setContentView(binding.*root*)

*window*.*statusBarColor* = Color.*WHITE*

val q = *intent*.getSerializableExtra(*EXTRA\_QUESTIONS*) as ModelQuestions

questions = q.questions

adapter = ResultsRecyclerViewAdapter()

countScore()

binding.recyclerView.*adapter* = adapter

adapter.submitList(questions)

binding.arrBack.setOnClickListener **{**

onBackPressed()

**}**

binding.doneButton.setOnClickListener **{**

onBackPressed()

**}**

binding.buttonDone.setOnClickListener **{**

onBackPressed()

**}**

}

private fun countScore() {

for (q in questions) {

if (q.userAnswer == q.answer)

score++

}

val result = "$score / ${questions.size}"

binding.resultScoreTextView.*text* = result

}

}