package com.twitter.follow\_recommendations.flows.post\_nux\_ml

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.follow\_recommendations.common.clients.dismiss\_store.DismissStore

import com.twitter.follow\_recommendations.common.clients.geoduck.UserLocationFetcher

import com.twitter.follow\_recommendations.common.clients.impression\_store.WtfImpressionStore

import com.twitter.follow\_recommendations.common.clients.interests\_service.InterestServiceClient

import com.twitter.follow\_recommendations.common.clients.socialgraph.SocialGraphClient

import com.twitter.follow\_recommendations.common.clients.user\_state.UserStateClient

import com.twitter.follow\_recommendations.common.predicates.dismiss.DismissedCandidatePredicateParams

import com.twitter.follow\_recommendations.common.utils.RescueWithStatsUtils.\_

import com.twitter.follow\_recommendations.flows.post\_nux\_ml.PostNuxMlRequestBuilderParams.DismissedIdScanBudget

import com.twitter.follow\_recommendations.flows.post\_nux\_ml.PostNuxMlRequestBuilderParams.TopicIdFetchBudget

import com.twitter.follow\_recommendations.flows.post\_nux\_ml.PostNuxMlRequestBuilderParams.WTFImpressionsScanBudget

import com.twitter.follow\_recommendations.products.common.ProductRequest

import com.twitter.inject.Logging

import com.twitter.stitch.Stitch

import com.twitter.util.Time

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class PostNuxMlRequestBuilder @Inject() (

socialGraph: SocialGraphClient,

wtfImpressionStore: WtfImpressionStore,

dismissStore: DismissStore,

userLocationFetcher: UserLocationFetcher,

interestServiceClient: InterestServiceClient,

userStateClient: UserStateClient,

statsReceiver: StatsReceiver)

extends Logging {

val stats: StatsReceiver = statsReceiver.scope("post\_nux\_ml\_request\_builder")

val invalidRelationshipUsersStats: StatsReceiver = stats.scope("invalidRelationshipUserIds")

private val invalidRelationshipUsersMaxSizeCounter =

invalidRelationshipUsersStats.counter("maxSize")

private val invalidRelationshipUsersNotMaxSizeCounter =

invalidRelationshipUsersStats.counter("notMaxSize")

def build(

req: ProductRequest,

previouslyRecommendedUserIds: Option[Set[Long]] = None,

previouslyFollowedUserIds: Option[Set[Long]] = None

): Stitch[PostNuxMlRequest] = {

val dl = req.recommendationRequest.displayLocation

val resultsStitch = Stitch.collect(

req.recommendationRequest.clientContext.userId

.map { userId =>

val lookBackDuration = req.params(DismissedCandidatePredicateParams.LookBackDuration)

val negativeStartTs = -(Time.now - lookBackDuration).inMillis

val recentFollowedUserIdsStitch =

rescueWithStats(

socialGraph.getRecentFollowedUserIds(userId),

stats,

"recentFollowedUserIds")

val invalidRelationshipUserIdsStitch =

if (req.params(PostNuxMlParams.EnableInvalidRelationshipPredicate)) {

rescueWithStats(

socialGraph

.getInvalidRelationshipUserIds(userId)

.onSuccess(ids =>

if (ids.size >= SocialGraphClient.MaxNumInvalidRelationship) {

invalidRelationshipUsersMaxSizeCounter.incr()

} else {

invalidRelationshipUsersNotMaxSizeCounter.incr()

}),

stats,

"invalidRelationshipUserIds"

)

} else {

Stitch.value(Seq.empty)

}

// recentFollowedByUserIds are only used in experiment candidate sources

val recentFollowedByUserIdsStitch = if (req.params(PostNuxMlParams.GetFollowersFromSgs)) {

rescueWithStats(

socialGraph.getRecentFollowedByUserIdsFromCachedColumn(userId),

stats,

"recentFollowedByUserIds")

} else Stitch.value(Seq.empty)

val wtfImpressionsStitch =

rescueWithStatsWithin(

wtfImpressionStore.get(userId, dl),

stats,

"wtfImpressions",

req.params(WTFImpressionsScanBudget))

val dismissedUserIdsStitch =

rescueWithStatsWithin(

dismissStore.get(userId, negativeStartTs, None),

stats,

"dismissedUserIds",

req.params(DismissedIdScanBudget))

val locationStitch =

rescueOptionalWithStats(

userLocationFetcher.getGeohashAndCountryCode(

Some(userId),

req.recommendationRequest.clientContext.ipAddress),

stats,

"userLocation"

)

val topicIdsStitch =

rescueWithStatsWithin(

interestServiceClient.fetchUttInterestIds(userId),

stats,

"topicIds",

req.params(TopicIdFetchBudget))

val userStateStitch =

rescueOptionalWithStats(userStateClient.getUserState(userId), stats, "userState")

Stitch.join(

recentFollowedUserIdsStitch,

invalidRelationshipUserIdsStitch,

recentFollowedByUserIdsStitch,

dismissedUserIdsStitch,

wtfImpressionsStitch,

locationStitch,

topicIdsStitch,

userStateStitch

)

})

resultsStitch.map {

case Some(

(

recentFollowedUserIds,

invalidRelationshipUserIds,

recentFollowedByUserIds,

dismissedUserIds,

wtfImpressions,

locationInfo,

topicIds,

userState)) =>

PostNuxMlRequest(

params = req.params,

clientContext = req.recommendationRequest.clientContext,

similarToUserIds = Nil,

inputExcludeUserIds = req.recommendationRequest.excludedIds.getOrElse(Nil),

recentFollowedUserIds = Some(recentFollowedUserIds),

invalidRelationshipUserIds = Some(invalidRelationshipUserIds.toSet),

recentFollowedByUserIds = Some(recentFollowedByUserIds),

dismissedUserIds = Some(dismissedUserIds),

displayLocation = dl,

maxResults = req.recommendationRequest.maxResults,

debugOptions = req.recommendationRequest.debugParams.flatMap(\_.debugOptions),

wtfImpressions = Some(wtfImpressions),

geohashAndCountryCode = locationInfo,

uttInterestIds = Some(topicIds),

inputPreviouslyRecommendedUserIds = previouslyRecommendedUserIds,

inputPreviouslyFollowedUserIds = previouslyFollowedUserIds,

isSoftUser = req.recommendationRequest.isSoftUser,

userState = userState

)

case \_ =>

PostNuxMlRequest(

params = req.params,

clientContext = req.recommendationRequest.clientContext,

similarToUserIds = Nil,

inputExcludeUserIds = req.recommendationRequest.excludedIds.getOrElse(Nil),

recentFollowedUserIds = None,

invalidRelationshipUserIds = None,

recentFollowedByUserIds = None,

dismissedUserIds = None,

displayLocation = dl,

maxResults = req.recommendationRequest.maxResults,

debugOptions = req.recommendationRequest.debugParams.flatMap(\_.debugOptions),

wtfImpressions = None,

geohashAndCountryCode = None,

inputPreviouslyRecommendedUserIds = previouslyRecommendedUserIds,

inputPreviouslyFollowedUserIds = previouslyFollowedUserIds,

isSoftUser = req.recommendationRequest.isSoftUser,

userState = None

)

}

}

}