package com.twitter.home\_mixer.product.list\_recommended\_users.feature\_hydrator

import com.twitter.home\_mixer.model.request.HasListId

import com.twitter.home\_mixer.product.list\_recommended\_users.model.ListRecommendedUsersFeatures.IsSGSValidUserFeature

import com.twitter.product\_mixer.component\_library.model.candidate.UserCandidate

import com.twitter.product\_mixer.core.feature.Feature

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMapBuilder

import com.twitter.product\_mixer.core.functional\_component.feature\_hydrator.BulkCandidateFeatureHydrator

import com.twitter.product\_mixer.core.model.common.CandidateWithFeatures

import com.twitter.product\_mixer.core.model.common.identifier.FeatureHydratorIdentifier

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.socialgraph.{thriftscala => sg}

import com.twitter.stitch.Stitch

import com.twitter.stitch.socialgraph.SocialGraph

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class IsSGSValidUserFeatureHydrator @Inject() (socialGraph: SocialGraph)

extends BulkCandidateFeatureHydrator[PipelineQuery with HasListId, UserCandidate] {

override val identifier: FeatureHydratorIdentifier =

FeatureHydratorIdentifier("IsSGSValidUser")

override def features: Set[Feature[\_, \_]] = Set(IsSGSValidUserFeature)

override def apply(

query: PipelineQuery with HasListId,

candidates: Seq[CandidateWithFeatures[UserCandidate]]

): Stitch[Seq[FeatureMap]] = {

val sourceId = query.getRequiredUserId

val targetUserIds = candidates.map(\_.candidate.id)

val request = sg.IdsRequest(

relationships = Seq(

sg.SrcRelationship(

source = sourceId,

relationshipType = sg.RelationshipType.Blocking,

hasRelationship = true,

targets = Some(targetUserIds)),

sg.SrcRelationship(

source = sourceId,

relationshipType = sg.RelationshipType.BlockedBy,

hasRelationship = true,

targets = Some(targetUserIds)),

sg.SrcRelationship(

source = sourceId,

relationshipType = sg.RelationshipType.Muting,

hasRelationship = true,

targets = Some(targetUserIds))

),

pageRequest = Some(sg.PageRequest(selectAll = Some(true))),

context = Some(sg.LookupContext(performUnion = Some(true)))

)

socialGraph.ids(request).map(\_.ids).map(\_.toSet).map { hasRelationshipUserIds =>

candidates.map { candidate =>

FeatureMapBuilder()

.add(IsSGSValidUserFeature, !hasRelationshipUserIds.contains(candidate.candidate.id))

.build()

}

}

}

}