package com.twitter.home\_mixer.product.list\_recommended\_users.filter

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

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

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

import com.twitter.product\_mixer.core.functional\_component.filter.Filter

import com.twitter.product\_mixer.core.functional\_component.filter.FilterResult

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

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

import com.twitter.stitch.Stitch

object PreviouslyServedUsersFilter extends Filter[ListRecommendedUsersQuery, UserCandidate] {

override val identifier: FilterIdentifier = FilterIdentifier("PreviouslyServedUsers")

override def apply(

query: ListRecommendedUsersQuery,

candidates: Seq[CandidateWithFeatures[UserCandidate]]

): Stitch[FilterResult[UserCandidate]] = {

val recentListMembers = query.features.map(\_.getOrElse(RecentListMembersFeature, Seq.empty))

val servedUserIds = query.pipelineCursor.map(\_.excludedIds)

val excludedUserIds = (recentListMembers.getOrElse(Seq.empty) ++

query.selectedUserIds.getOrElse(Seq.empty) ++

query.excludedUserIds.getOrElse(Seq.empty) ++

servedUserIds.getOrElse(Seq.empty)).toSet

val (removed, kept) =

candidates.map(\_.candidate).partition(candidate => excludedUserIds.contains(candidate.id))

Stitch.value(FilterResult(kept = kept, removed = removed))

}

}