package com.twitter.follow\_recommendations.common.stores

import com.twitter.follow\_recommendations.common.models.CandidateUser

import com.twitter.follow\_recommendations.common.models.HasRecentFollowedUserIds

import com.twitter.stitch.Stitch

import com.twitter.strato.generated.client.onboarding.userrecs.TweepCredOnUserClientColumn

import javax.inject.Inject

import javax.inject.Singleton

// Not a candidate source since it's a intermediary.

@Singleton

class LowTweepCredFollowStore @Inject() (tweepCredOnUserClientColumn: TweepCredOnUserClientColumn) {

def getLowTweepCredUsers(target: HasRecentFollowedUserIds): Stitch[Seq[CandidateUser]] = {

val newFollowings =

target.recentFollowedUserIds.getOrElse(Nil).take(LowTweepCredFollowStore.NumFlockToRetrieve)

val validTweepScoreUserIdsStitch: Stitch[Seq[Long]] = Stitch

.traverse(newFollowings) { newFollowingUserId =>

val tweepCredScoreOptStitch = tweepCredOnUserClientColumn.fetcher

.fetch(newFollowingUserId)

.map(\_.v)

tweepCredScoreOptStitch.map(\_.flatMap(tweepCred =>

if (tweepCred < LowTweepCredFollowStore.TweepCredThreshold) {

Some(newFollowingUserId)

} else {

None

}))

}.map(\_.flatten)

validTweepScoreUserIdsStitch

.map(\_.map(CandidateUser(\_, Some(CandidateUser.DefaultCandidateScore))))

}

}

object LowTweepCredFollowStore {

val NumFlockToRetrieve = 500

val TweepCredThreshold = 40

}