package com.twitter.frigate.pushservice.model.ntab

import com.twitter.frigate.common.base.SocialContextActions

import com.twitter.frigate.common.base.SocialContextUserDetails

import com.twitter.frigate.pushservice.model.PushTypes.PushCandidate

import com.twitter.frigate.pushservice.util.CandidateUtil

import com.twitter.util.Future

trait NTabSocialContext {

self: PushCandidate with SocialContextActions with SocialContextUserDetails =>

private def ntabDisplayUserIds: Seq[Long] =

socialContextUserIds.take(ntabDisplayUserIdsLength)

def ntabDisplayUserIdsLength: Int =

if (socialContextUserIds.size == 2) 2 else 1

def ntabDisplayNamesAndIds: Future[Seq[(String, Long)]] =

scUserMap.map { userObjMap =>

ntabDisplayUserIds.flatMap { id =>

userObjMap(id).flatMap(\_.profile.map(\_.name)).map { name => (name, id) }

}

}

def rankedNtabDisplayNamesAndIds(defaultToRecency: Boolean): Future[Seq[(String, Long)]] =

scUserMap.flatMap { userObjMap =>

val rankedSocialContextActivityFut =

CandidateUtil.getRankedSocialContext(

socialContextActions,

target.seedsWithWeight,

defaultToRecency)

rankedSocialContextActivityFut.map { rankedSocialContextActivity =>

val ntabDisplayUserIds =

rankedSocialContextActivity.map(\_.userId).take(ntabDisplayUserIdsLength)

ntabDisplayUserIds.flatMap { id =>

userObjMap(id).flatMap(\_.profile.map(\_.name)).map { name => (name, id) }

}

}

}

def otherCount: Future[Int] =

ntabDisplayNamesAndIds.map {

case namesWithIdSeq =>

Math.max(0, socialContextUserIds.length - namesWithIdSeq.size)

}

}