package com.twitter.unified\_user\_actions.adapter.client\_event

import com.twitter.clientapp.thriftscala.ItemType

import com.twitter.clientapp.thriftscala.LogEvent

import com.twitter.clientapp.thriftscala.{Item => LogEventItem}

import com.twitter.unified\_user\_actions.thriftscala.\_

abstract class BaseFeedbackSubmitClientEvent(actionType: ActionType)

extends BaseClientEvent(actionType = actionType) {

override def getUuaItem(

ceItem: LogEventItem,

logEvent: LogEvent

): Option[Item] = {

logEvent.eventNamespace.flatMap(\_.page).flatMap {

case "search" =>

val searchInfoUtil = new SearchInfoUtils(ceItem)

searchInfoUtil.getQueryOptFromItem(logEvent).flatMap { query =>

val isRelevant: Boolean = logEvent.eventNamespace

.flatMap(\_.element)

.contains("is\_relevant")

logEvent.eventNamespace.flatMap(\_.component).flatMap {

case "relevance\_prompt\_module" =>

for (actionTweetId <- ceItem.id)

yield Item.FeedbackPromptInfo(

FeedbackPromptInfo(

feedbackPromptActionInfo = FeedbackPromptActionInfo.TweetRelevantToSearch(

TweetRelevantToSearch(

searchQuery = query,

tweetId = actionTweetId,

isRelevant = Some(isRelevant)))))

case "did\_you\_find\_it\_module" =>

Some(

Item.FeedbackPromptInfo(FeedbackPromptInfo(feedbackPromptActionInfo =

FeedbackPromptActionInfo.DidYouFindItSearch(

DidYouFindItSearch(searchQuery = query, isRelevant = Some(isRelevant))))))

}

}

case \_ => None

}

}

override def isItemTypeValid(itemTypeOpt: Option[ItemType]): Boolean =

ItemTypeFilterPredicates.isItemTypeForSearchResultsPageFeedbackSubmit(itemTypeOpt)

}