package com.twitter.home\_mixer.functional\_component.side\_effect

import com.twitter.home\_mixer.model.HomeFeatures.ImpressionBloomFilterFeature

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

import com.twitter.home\_mixer.param.HomeGlobalParams.EnableImpressionBloomFilter

import com.twitter.home\_mixer.service.HomeMixerAlertConfig

import com.twitter.product\_mixer.core.functional\_component.side\_effect.PipelineResultSideEffect

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

import com.twitter.product\_mixer.core.model.common.presentation.CandidateWithDetails

import com.twitter.product\_mixer.core.model.marshalling.HasMarshalling

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

import com.twitter.stitch.Stitch

import com.twitter.timelines.clients.manhattan.store.ManhattanStoreClient

import com.twitter.timelines.impressionbloomfilter.{thriftscala => blm}

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class PublishImpressionBloomFilterSideEffect @Inject() (

bloomFilterClient: ManhattanStoreClient[

blm.ImpressionBloomFilterKey,

blm.ImpressionBloomFilterSeq

]) extends PipelineResultSideEffect[PipelineQuery with HasSeenTweetIds, HasMarshalling]

with PipelineResultSideEffect.Conditionally[

PipelineQuery with HasSeenTweetIds,

HasMarshalling

] {

override val identifier: SideEffectIdentifier =

SideEffectIdentifier("PublishImpressionBloomFilter")

private val SurfaceArea = blm.SurfaceArea.HomeTimeline

override def onlyIf(

query: PipelineQuery with HasSeenTweetIds,

selectedCandidates: Seq[CandidateWithDetails],

remainingCandidates: Seq[CandidateWithDetails],

droppedCandidates: Seq[CandidateWithDetails],

response: HasMarshalling

): Boolean =

query.params.getBoolean(EnableImpressionBloomFilter) && query.seenTweetIds.exists(\_.nonEmpty)

def buildEvents(query: PipelineQuery): Option[blm.ImpressionBloomFilterSeq] = {

query.features.flatMap { featureMap =>

val impressionBloomFilterSeq = featureMap.get(ImpressionBloomFilterFeature)

if (impressionBloomFilterSeq.entries.nonEmpty) Some(impressionBloomFilterSeq)

else None

}

}

override def apply(

inputs: PipelineResultSideEffect.Inputs[PipelineQuery with HasSeenTweetIds, HasMarshalling]

): Stitch[Unit] = {

buildEvents(inputs.query)

.map { updatedBloomFilterSeq =>

bloomFilterClient.write(

blm.ImpressionBloomFilterKey(inputs.query.getRequiredUserId, SurfaceArea),

updatedBloomFilterSeq)

}.getOrElse(Stitch.Unit)

}

override val alerts = Seq(

HomeMixerAlertConfig.BusinessHours.defaultSuccessRateAlert(99.8)

)

}