package com.twitter.home\_mixer.functional\_component.gate

import com.twitter.conversions.DurationOps.\_

import com.twitter.product\_mixer.core.feature.Feature

import com.twitter.product\_mixer.core.functional\_component.gate.Gate

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

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

import com.twitter.stitch.Stitch

import com.twitter.timelinemixer.clients.manhattan.DismissInfo

import com.twitter.timelineservice.suggests.thriftscala.SuggestType

import com.twitter.util.Duration

object DismissFatigueGate {

// how long a dismiss action from user needs to be respected

val DefaultBaseDismissDuration = 7.days

val MaximumDismissalCountMultiplier = 4

}

case class DismissFatigueGate(

suggestType: SuggestType,

dismissInfoFeature: Feature[PipelineQuery, Map[SuggestType, Option[DismissInfo]]],

baseDismissDuration: Duration = DismissFatigueGate.DefaultBaseDismissDuration,

) extends Gate[PipelineQuery] {

override val identifier: GateIdentifier = GateIdentifier("DismissFatigue")

override def shouldContinue(query: PipelineQuery): Stitch[Boolean] = {

val dismissInfoMap = query.features.map(

\_.getOrElse(dismissInfoFeature, Map.empty[SuggestType, Option[DismissInfo]]))

val isVisible = dismissInfoMap

.flatMap(\_.get(suggestType))

.flatMap(\_.map { info =>

val currentDismissalDuration = query.queryTime.since(info.lastDismissed)

val targetDismissalDuration = dismissDurationForCount(info.count, baseDismissDuration)

currentDismissalDuration > targetDismissalDuration

}).getOrElse(true)

Stitch.value(isVisible)

}

private def dismissDurationForCount(

dismissCount: Int,

dismissDuration: Duration

): Duration =

// limit to maximum dismissal duration

dismissDuration \* Math.min(dismissCount, DismissFatigueGate.MaximumDismissalCountMultiplier)

}