Observation for Class **MobileTrackingUtils**

**Reason for Low Coverage(92.8%) –**

**There are number of issues-**

1. **Methods- Below are the methods having same issue-**
2. **trackClarify**(final HttpServletRequest request, final SearchParameters searchParameters){..}
3. **trackCommonElementFallBack**(final HttpServletRequest request){..}
4. **trackCommonElemOnPages**(final HttpServletRequest request){..}
5. **trackEnrollment**(final HttpServletRequest request){..}
6. **trackHome**(final HttpServletRequest request){..}
7. **trackMyReservationDetails**(final HttpServletRequest request){..}
8. **trackSearch**(final HttpServletRequest request){..}

**Issue : As per logic defined-**

public void trackSearch(final HttpServletRequest request) {

try {

Map<String, Object> tmsMap = new HashMap<String, Object>();

// Common Elements

this.trackCommonElements(request, tmsMap);

// Search Term on Clarify Page

tmsMap.put(MobileTrackingConstants.PAGE\_ID, MobileTrackingConstants.PAGE\_ID\_SEARCH);

// Setting the JSON to request

this.setTMSAttribute(request, tmsMap);

} catch (Exception exception) {

LOGGER.error("Exception at trackSearch method of MobileTrackingUtils ", exception);

}

}

**unnecessary use of try..catch block**, Since exception is already handled in following method

* trackCommonElements(request, tmsMap),
* setTMSAttribute(request, tmsMap);

hence, exception will never be thrown by these method.

**Proposed Approach: Remove try..catch block, since exception is already catched to their respective method called by these method.**

**-----------------------------------------------------------------------------------**

1. **Method-** trackCommonElements(final HttpServletRequest request, final Map<String, Object> tmsMap)

**Issue :** Refer code – 454-456

if (null != MobileTrackingConstants.IATA\_NUMBER) {

tmsMap.put(MobileTrackingConstants.IATA, request.getAttribute(MobileTrackingConstants.IATA\_NUMBER));

}

IATA\_NUMBER - Because it is declared as final in MobileTrackingConstants class, we cannot change its value. Hence unnecessary check is used.

**Proposed Approach: Remove the above check.**

**------------------------------------------------------------------------------------**

1. **Method-** private static String **getHotels**(List<HotelResultItem> hotels) **🡪** called by trackSearchResults(…)

**Issue: As per logic defined-** Refer 680-684, 107

In Method trackSearchResults(…)

**if** (**null** != hotels && !hotels.isEmpty()) {

tmsMap.put(MobileTrackingConstants.*NUM\_OF\_HOTELS*, **this**.getJSONString(hotels.size()));

tmsMap.put(MobileTrackingConstants.*HOTEL\_ID*, ***getHotels***(hotels));

}

In Method **private** **static** String getHotels(List<HotelResultItem> hotels) {

**if** (hotels != **null**){

..}

As we can see, redundant check is used in getHotels(.) method, hence it will never be null.

**Proposed Approach: Remove the redundant check used in getHotels(.) method**

**-----------------------------------------------------------------------------------**

1. **Method- private** **void** **fillRoomDetailInfo(Map<String, Object> tmsMap, RateDetail rateDetail, int totalRoomNights)** 🡪 called by trackBookingConfirmation(..)

**Issue :** Refer code – 228-232

In Method **fillRoomDetailInfo** (…)

if (null != rateDetail && CollectionUtils.isNotEmpty(rateDetail.getRoomBeans())) {

Double taxUSD = 0.0;

Double totalPriceUSD = 0.0;

RoomDetail roomDetail = rateDetail.getRoomBeans().get(0);

if (null != roomDetail) {

…}

}

Redundant check is used for roomDetail, since in a first condition rateDetail.getRoomBeans list will never be empty, hence roomDetail cannot be null.

**Proposed Approach: Remove the redundant check (null != roomDetail) used in above method.**