DESIGN DOCUMENT

Collaborators:

|  |  |  |
| --- | --- | --- |
| Project document | Version | Owner |
| Initial document | 1.0 | Deepakraj Kanchipuram Swami |

Contents

[1. Introduction 4](#_Toc65812291)

[2. Purpose 4](#_Toc65812292)

[3. Project Scope 4](#_Toc65812293)

[a. Inscope: 4](#_Toc65812294)

[b. OutScope: 4](#_Toc65812295)

[4. System requirements and Design: 4](#_Toc65812296)

[a. Hardware: 4](#_Toc65812297)

[b. Software: 5](#_Toc65812298)

[c. Input: 5](#_Toc65812299)

[d. Output: 5](#_Toc65812300)

[e. Logic: 5](#_Toc65812301)

[f. Codes: 6](#_Toc65812302)

[g. Constraints: 6](#_Toc65812303)

[5. Unit Testing: 6](#_Toc65812304)

[6. System Security: 7](#_Toc65812305)

[7. Portability: 7](#_Toc65812306)

# Introduction

This project mainly deals with identifying user consumption across our platform. This project aims to identify the first and recent activities that the user undertook on our platform. The data produced will be analyzed and used by other teams for further processing. This data can further be used in conjunction with registration details to identify the top age group accessing our platform, peak time, and other attributes to make informed decisions.

# Purpose

The purpose of this project is to leverage the use of Kafka in real-time streaming which will store the incoming stream of data produced by a Kafka producer, in a topic named “input\_topic”. The above-produced input\_topic will be consumed by Kafka Streams application to perform aggregation to identify the first and last visit by a particular consumer in real-time which will be written to a topic named “output\_topic”.

# Project Scope

## Inscope:

* + 1. Produce message and publish to Kafka topic (input\_topic):
    2. Consume (read) messages from the topic (input topic)
    3. Aggregate the messages by userId and produce a summary item to another Kafka topic (output\_topic)

## OutScope:

* + 1. Data analysis
    2. Reporting and Dashboarding

# System requirements and Design:

The tools used for implementing the project are explained below.

## Hardware:

The hardware used for writing the code and implementing the same is as follows.

* + 1. Windows 10
    2. Java 8
    3. Kafka 2.12 – 2.50
    4. IntelliJ IDE
    5. Maven (for loading dependencies and managing projects)

## Software:

The packages and dependencies used for implementing the project are as follows.

* + 1. Kafka Streams
    2. Slf4J – for logging purposes
    3. Maven assembly plugins – for creating jar with dependencies

## Input:

The producer should create JSON objects and load them into the input topic. The JSON messages should follow the below format.

{

"userId":"userid-1",

"type":"event",

"metadata": {

"messageId":"123sfdafas-32487239857dsh98234",

"sentAt":1534382478,

"timestamp":1534382478,

"receivedAt":0,

"apiKey":"apikey1",

"spaceId":"space1",

"version":"v1"

},

"event":"Played Movie",

"eventData":{

"MovieID":"MIM4ddd4"

}

}

## Output:

The output\_topic should contain the aggregated messages in the below format.

{

"userId":"j11288090",

"firstSeen":1534382478,

"lastSeen":1534386588

}

## Logic:

The logic used for implementing the same is explained below. From the output, it is evident that,

* + 1. The field ‘firstSeen’ represents A user's earliest record/ user's time when he saw their first ever movie
    2. The field ‘lastSeen’ represents A user's latest record/ user's time when he saw the latest movie

An aggregator is used to find the minimum timestamp and maximum timestamp between the incoming value and the intermediate value thereby populating the values for the fields ‘firstSeen’ and ‘lastSeen’ respectively. This in turn will be populated inside an empty JSON object along with the key (userId) and will be sent to the output topic.

## Codes:

The jar files used for the projects are as follows.

* + 1. ~KafkaJson\Producer\ New-1.0-SNAPSHOT-jar-with-dependencies
    2. ~KafkaJson\Streams\ New-1.0-SNAPSHOT-jar-with-dependencies
    3. ~KafkaJson\Unit Testing\ New-1.0-SNAPSHOT-jar-with-dependencies

The instructions on how to run the code and perform manual testing are mentioned in the README file.

## Constraints:

There are no design constraints

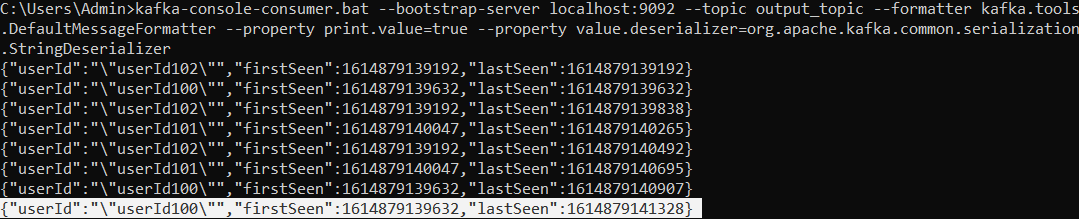
All the codes are created as per the design principle following security standards.

# Unit Testing:

Unit testing is performed, and the sample test cases are as follows.

Input\_topic (messages sent by the producer):

It is evident from the graph that the first occurrence of ‘userId100’ is at 161489139632 and its last occurrence is recorded at 1614879141328.

Output\_topic (Streams result):

The resulting output shows the same.

# System Security:

The code will run on JVM which is highly secure due to its non-pointer concept and secure platform build.

# Portability:

Java supports write once and execute anywhere concept as it is platform-independent. Hence the jars can be executed anywhere with Java 8 installed.