```
package com.prodapt.hack
import org.apache.spark.SparkContext
import org.apache.spark.SparkConf
import org.apache.spark.sql.types.{ StructType, StructField,StringType}

object prodapt2 {
def main(args: Array[String]) {
        println("******initialize the entry point, spark session to enter into spark world
******")

/*Define the spark config*/
  val spark=org.apache.spark.sql.SparkSession.builder()
            .appName("hackathon prodapt")
            .config("spark.history.fs.logDirectory", "file:///tmp/spark-events")
            .config("spark.eventLog.dir", "file:////tmp/spark-events")
            .config("spark.eventLog.enabled", "true").getOrCreate()

/* Create sc by invoking spark context class*/
  val sc=spark.sparkContext

/*Set the logger level*/
  sc.setLogLevel("error")

/*Create sql context*/
  val sqlc=spark.sqlContext

/* Set the shuffle partition to 1 instead of default value of 200 */
  spark.sqlContext.setConf("spark.sql.shuffle.partitions","1")

/* Define the struct type based on source file input.txt*/
  val struct_data = StructType(Array(StructField("message", StringType, true)))

/* Take the source file path as argument and read using readstream */
  val stream=spark.readStream.schema(struct_data).json(args(0))

/* Filter the message not null data and replace the whitespace with ~ */
  val filter_data=stream.where("message is not null")
                  .withColumn("message_split",regexp_replace(col("message"),"\\s+","~"))
                  .select(col("message_split"))

/* Fetch individual columns as per the output.txt */
  val column_split=filtered_data
                  .withColumn("split2",split(col("message_split"),"~"))
                  .select(col("split2").getItem(1).as("date"),
                   col("split2").getItem(2).as("timestamp"),
                   col("split2").getItem(12).as("URL"))
                   .where("URL like 'http://omwssu%'")
/*Data processing*/
   val processed_data=column_split
                    .select(substring_index(col("URL"),"/",4).as("fqdn"),
                     substring_index(substring_index(col("URL"),"/",5),"/",-1).as("cpe_id"),
                     substring_index(substring_index(col("URL"),"/",6),"/",-1).as("action"),
                     concat(substring_index(substring_index(col("URL"),"/",7),"/",-1),

lit("."),substring_index(substring_index(col("URL"),"/",8),"/",-1)).as("error_code"),
                     col("URL").as("message"),
                     concat(col("date"),lit(" "),col("timestamp")).as("timestamp"))

/* Write the data by passing target directory as argument and by using writestream*/
    val output = processed_data
                      .writeStream
                      .format("json")
                     .option("path",args(1))
                     .option("checkpointLocation", "/user/checkpoint")
                     .outputMode("append")
                     .start()
                     .awaitTermination()
  }

}
```