Technical Challenge Instructions:

1. Write a Golang Kafka Publisher that takes a CSV file and published each line as a message to a topic with 4 partitions.
2. Write a Golang Kafka Consumer that uses a consumer group to read off the topic for any message that calculates the sum of all the numbers and output the sum, read off the topic for any messages with English words and sorts the words alphabetically and outputs.
3. Show a high-level design document of the application and how it follows a RESTful design pattern.
4. Use *go lint*, *go vet*, and *go test* for testing the application.
5. Demo of the application taking the provided CSV file.

Solution:

1. The *Golang Kafka Publisher* is a RESTful Go application having a *POST* REST endpoint (*localhost:9090/csv*) that takes a JSON payload having a *location* field that is the location of the CSV file to be retrieved and published as messages to the *testTopic* topic in the Kafka cluster.
2. The *Golang Kafka Consumer* is a Go application that is a part of the *group-id-1* consumer group and consumes messages published to the *testTopic* topic in the Kafka cluster. The consumer application performs a check on the message to determine if the line contains numbers or strings values. It adds the numbers together and prints out the sum to the console if it contains numbers, otherwise it sorts the strings and prints out the sorted values to the console.
3. High-Level Design of the Architecture.

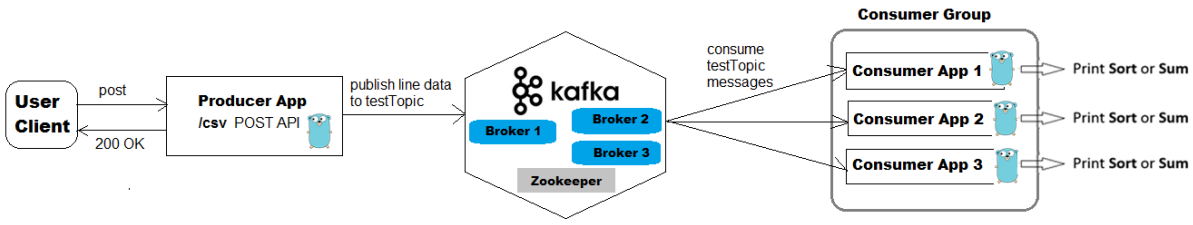

Fig 1: High-level Architecture

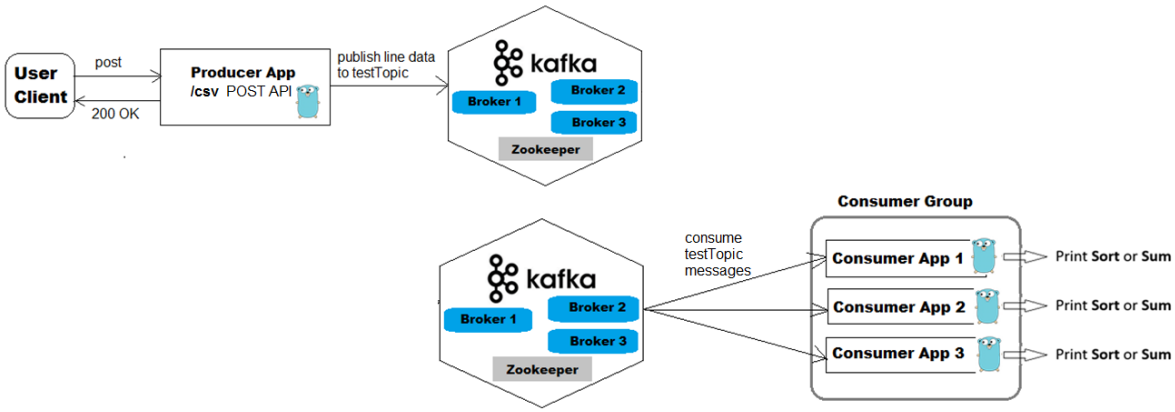

Fig 2: Decoupled High-level Architecture showing separate message publishing and consumption views

- The Kafka cluster consists of 3 brokers.
- A topic called testTopic having 4 partitions with a replication factor of 3. It was created using the *kafka-tool* as Fig 7.
- Several other figures are provided below that illustrate how to create and start the Zookeeper and Kafka clusters, the Kafka brokers, the Producer and Consumer applications.
- The Producer POST RESTful endpoint at *localhost:9090/csv* was triggered with Postman client.

- Sample output of the raw CSV data and the console outputs of the processed Consumer application results.
- The development environment was configured to use *go lint* and *go vet* by running the commands *go get -u golang.org/x/lint/golint* and *go tool vet *.go* in the respective project directories.
- The Unit testing file *consumer_test.go* is shown in Fig 15 and the unit test results for testing the Consumer application are then displayed in Fig 16.

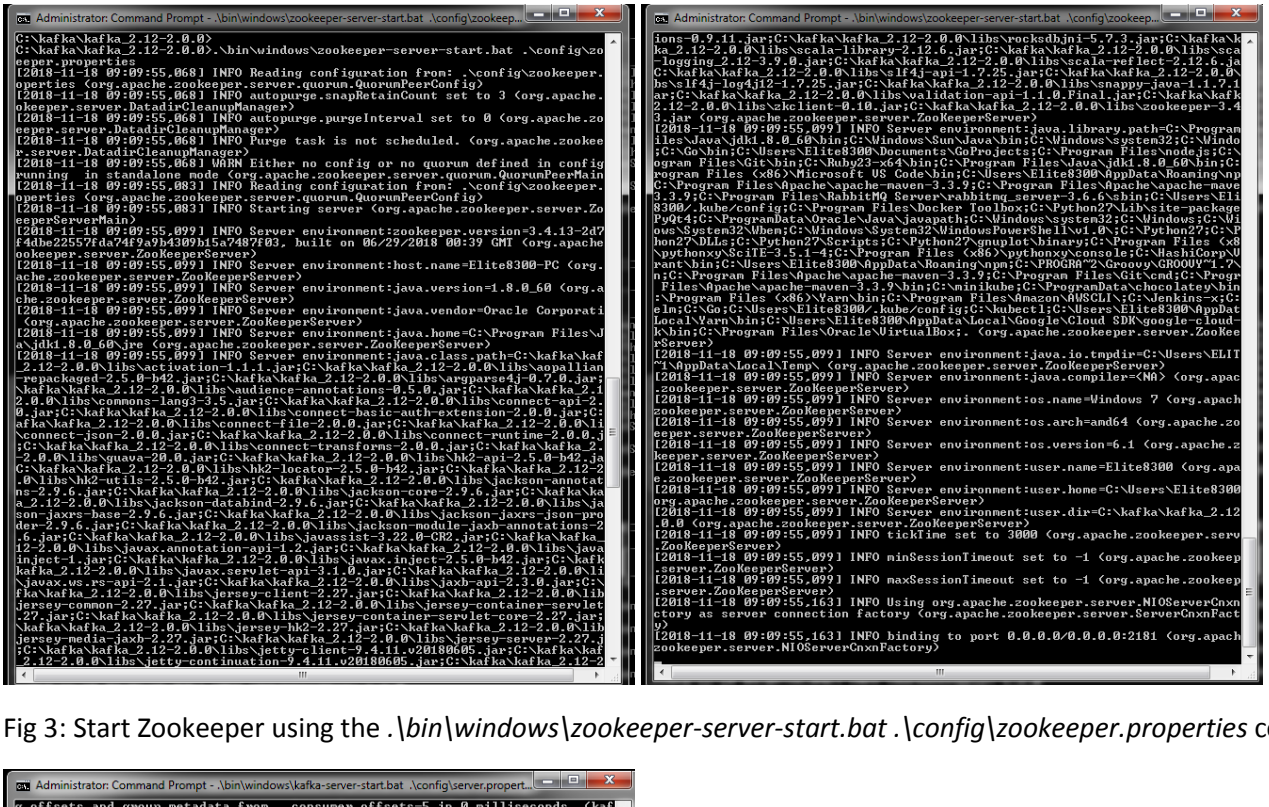

Fig 3: Start Zookeeper using the *.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties* command

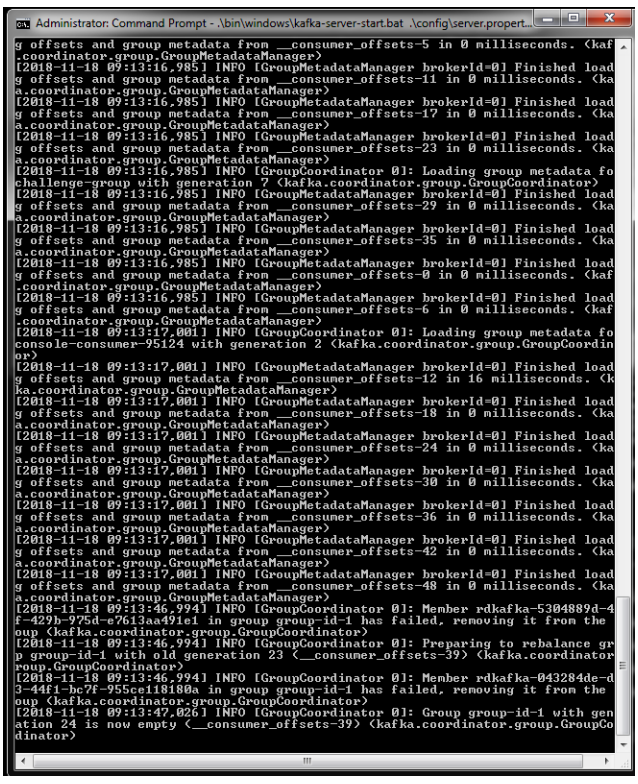

Fig 4: Start Kafka broker 1 as brokerId=0 using *.\bin\windows\kafka-server-start.bat .\config\server-1.properties* command

Fig 5: Start Kafka broker 2 as brokerId=1 using *.\bin\windows\kafka-server-start.bat .\config\server-2.properties* command

Fig 6: Start Kafka broker 3 as brokerId=2 using *.\bin\windows\kafka-server-start.bat .\config\server-3.properties* command

Fig 7: Create a new topic *testTopic* using the *.\bin\windows\kafka-topics.bat - -create - -zookeeper localhost:2181 - -replication-factor 3 - -partitions 4 - -topic testTopic* command

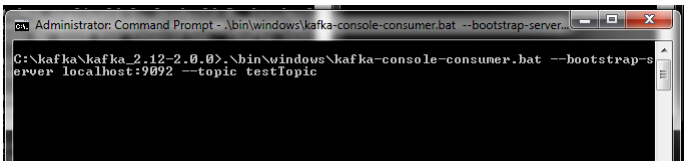

Fig 8: Create a console consumer for the *testTopic* topic using the *.\bin\windows\kafka-console-consumer.bat - -bootstrap-server localhost:9092 - -topic testTopic* command

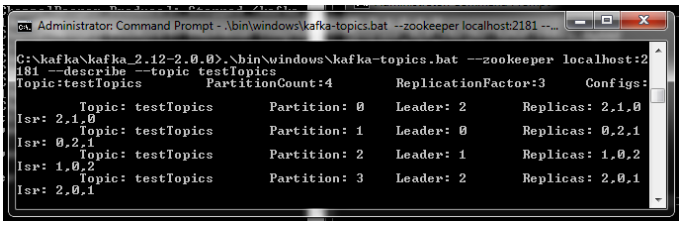

Fig 9: View partition distribution across brokers using the *.\bin\windows\kafka-topics.bat - -zookeeper localhost:2181 - -describe - -topic testTopics* command

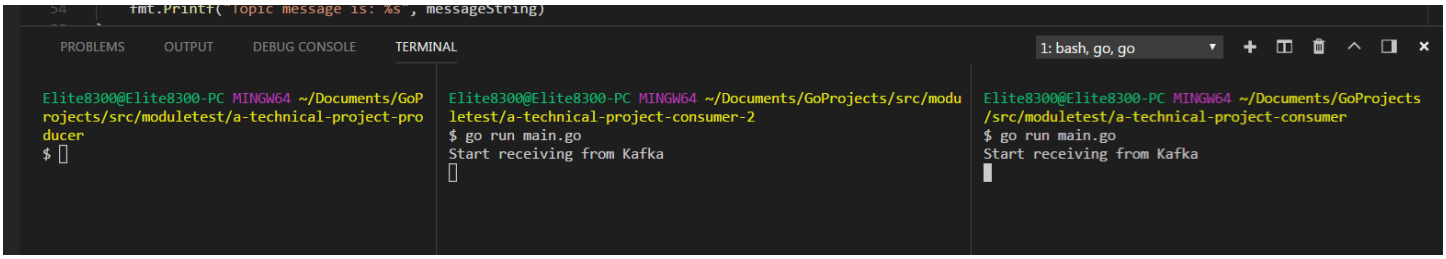

Fig 10: Starting the 2 consumers in the same consumer group for the *testTopic* topic

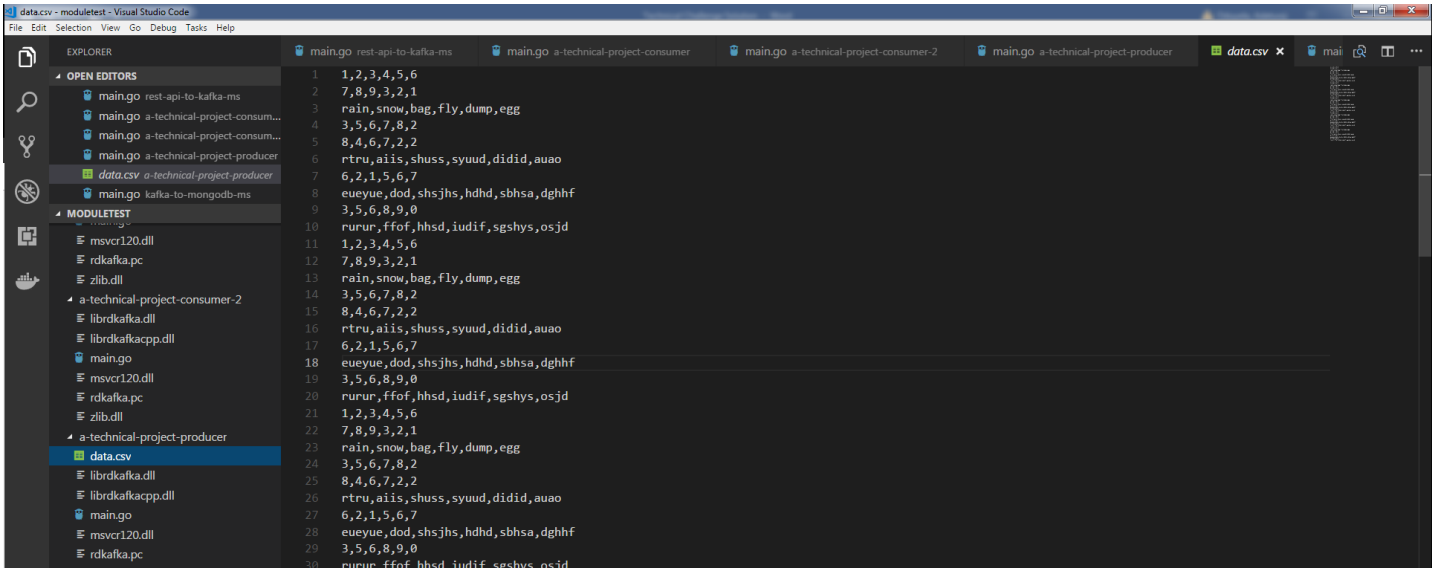

Fig 11: CSV file to be read by producer on POST request and sent to Kafka cluster *testTopic* topic

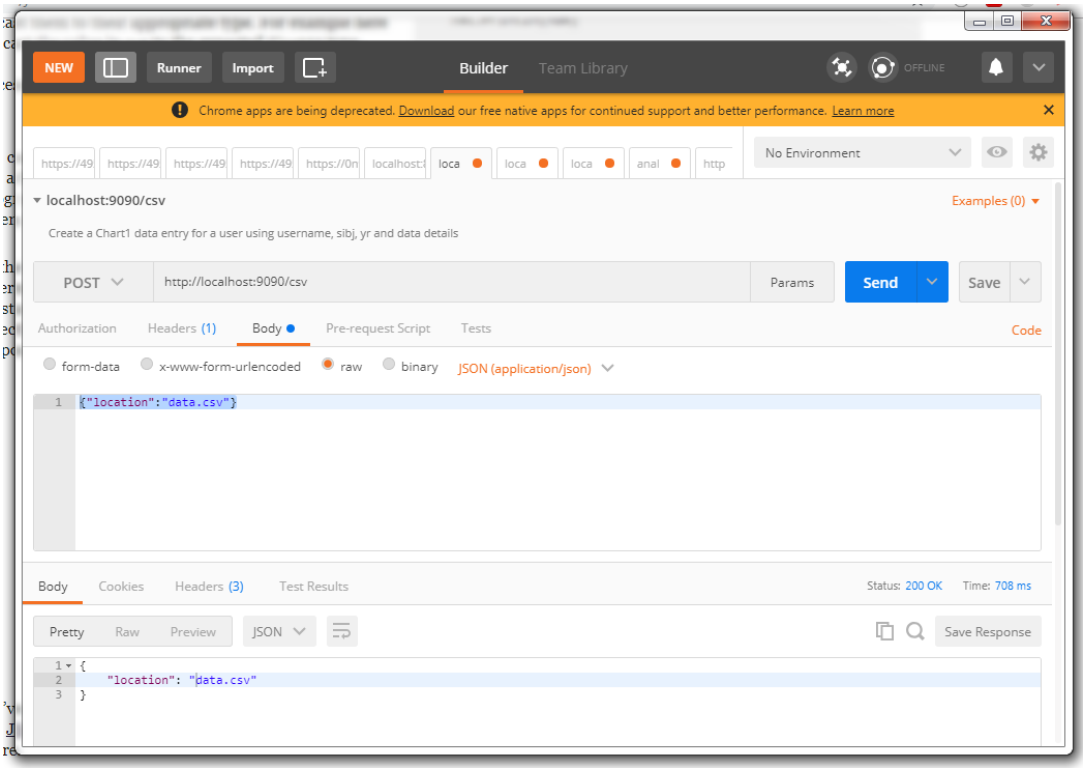

Fig 12: Producer POST API showing CSV file location to be retrieved and sent to Kafka broker as messages

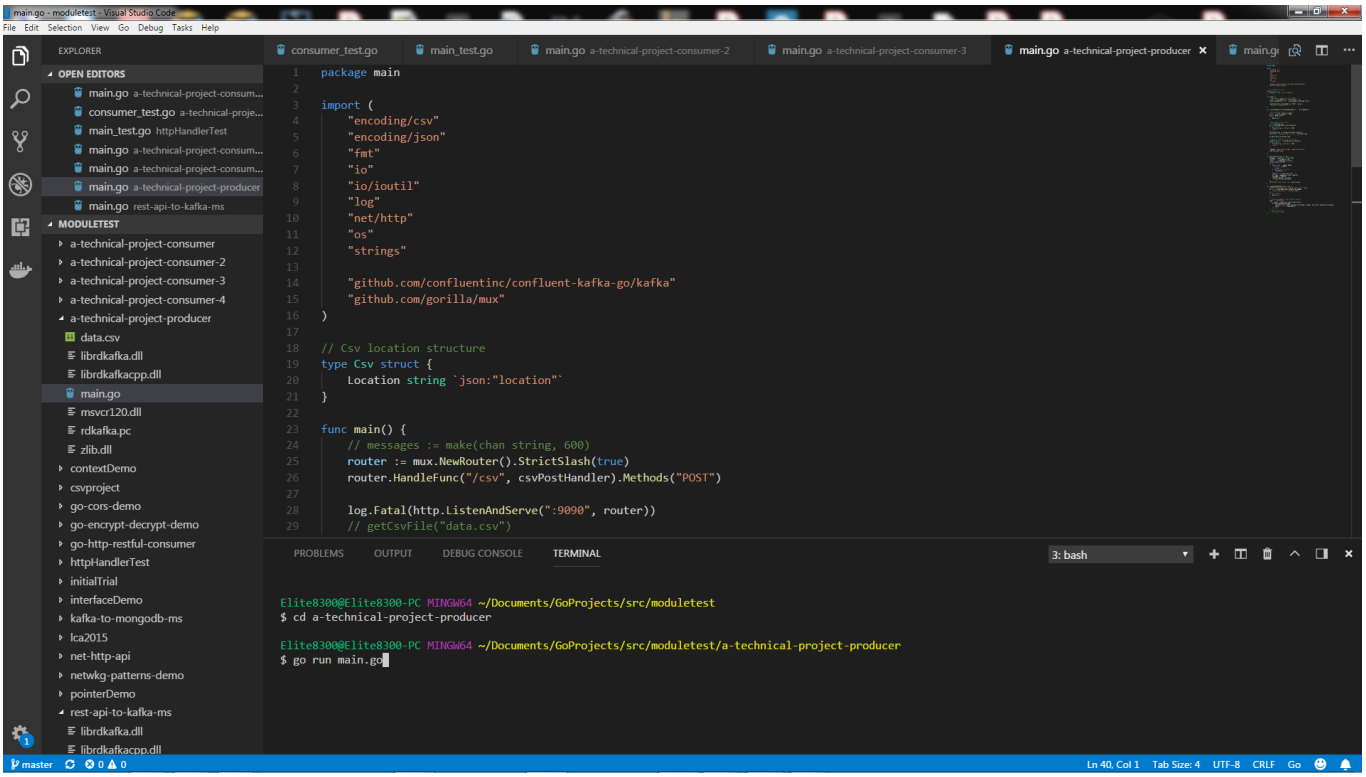

Fig 13: Producer code *main.go* file that takes a POST call with CSV file location and sends the CSV data lines as messages to the *testTopic* topic

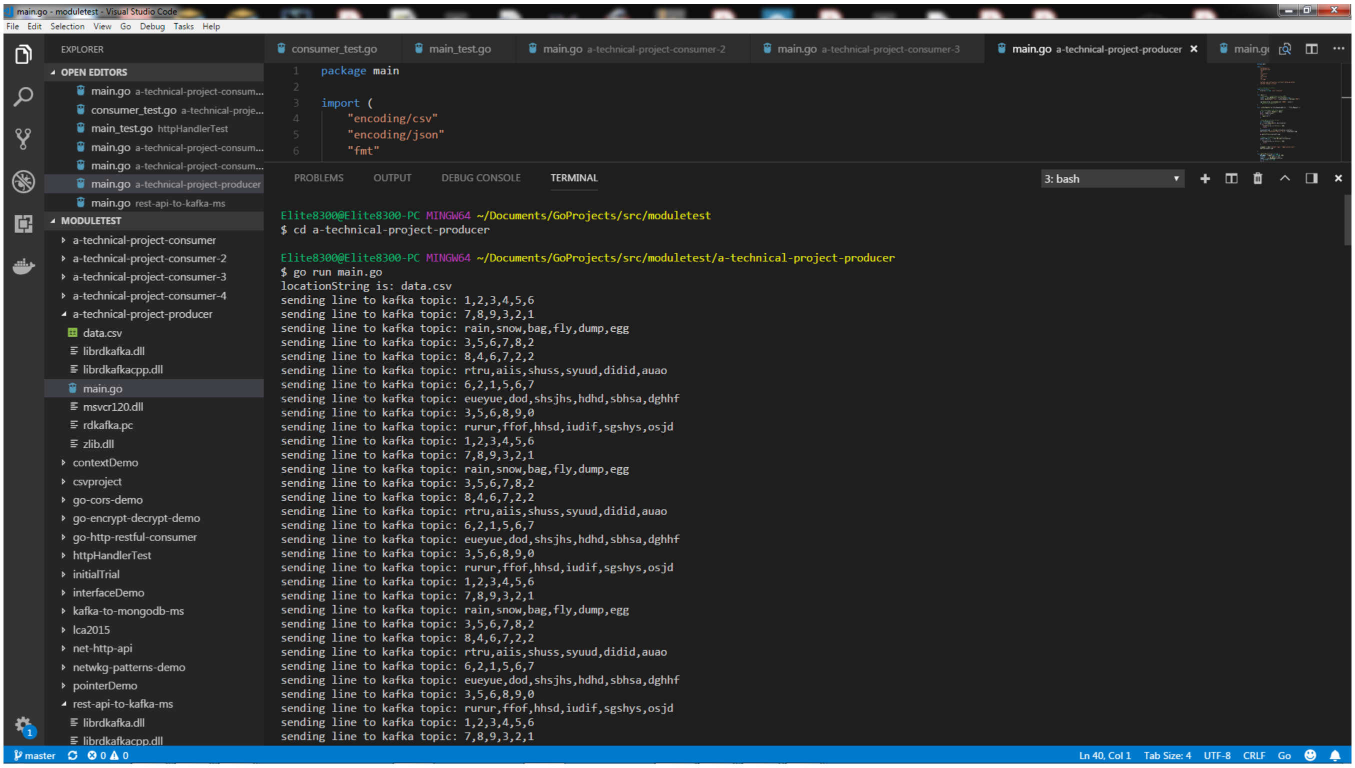

Fig 14: Sample Console log of messages from CSV file after POST call to Producer API

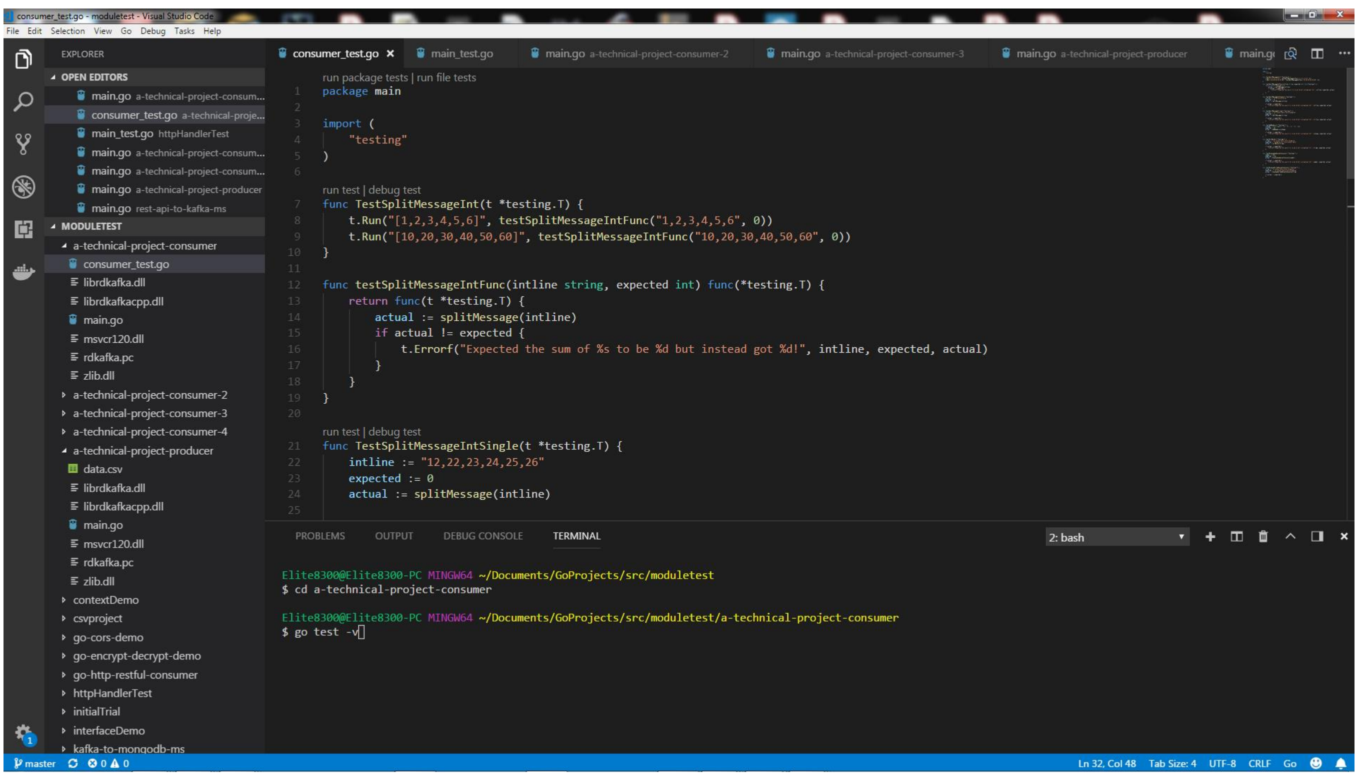

Fig 15: Unit Test file *consumer_test.go* for the consumer app

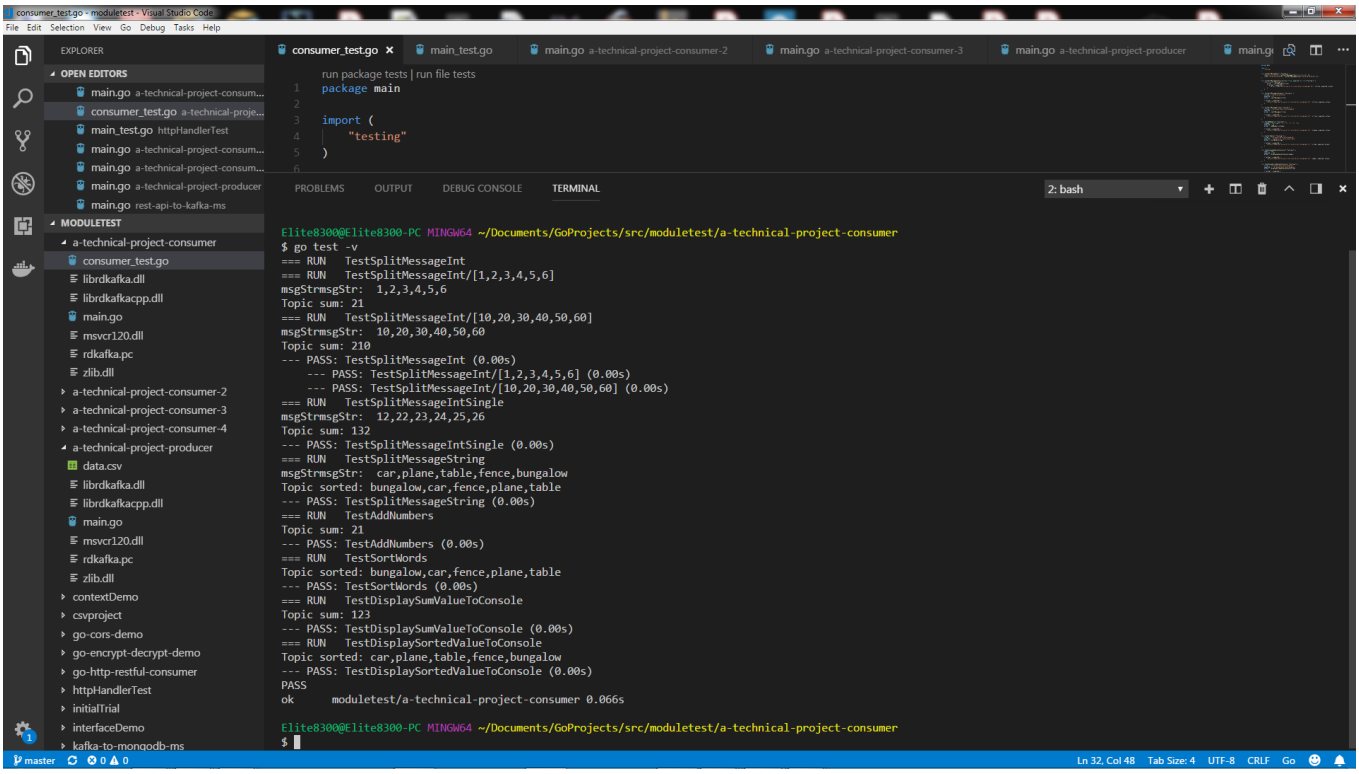

Fig 16: Unit test results for consumer application.